2017 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing

Parallel execution of cellular automata through space partitioning: the landslide simulation SciddicaS3-hex case study

Andrea Giordano*, Alessio De Rango†, Davide Spataro†,
Donato D'Ambrosio†, Carlo Mastroianni*, Gianluigi Folino*, William Spataro†
*ICAR-CNR, Rende, Italy
†University of Calabria, Department of Mathematics and Computer Science, Italy

***Abstract*—The performance and scalability of cellular automata, when executed on parallel/distributed machines, are limited by the necessity of synchronizing all the nodes at each time step, i.e., a node can execute its code only after all the other nodes have executed the previous step. However, if the code is parallelized by partitioning the space of the automata, these synchronization requirements can be relaxed: indeed, a node that manages a given portion of the cellular automata can execute a new step after synchronizing only with the nodes that manage the adjacent portions, while the remaining nodes can execute different time steps. This can be a notable advantage in many novel and increasingly popular applications of cellular automata, such as smart city applications, simulation of natural phenomena, etc., in which the execution times can be different and variable, due to the heterogeneity of machines and/or of the data and/or of the different functions. Indeed, a longer execution time at a node does not slow down the execution at all the other nodes but only at the neighboring nodes. This is particularly advantageous when the nodes that act as a bottleneck can vary during the execution. The goal of the paper is to analyze the benefits that can be achieved with the described approach when different space partitioning strategies are taken into account: i.e., the mono- and the two-dimensional partitioning. Experiments referred to a well-known cellular automata, namely the SciddicaS3-hex model for landslide simulation, exhibit good scalability and prove that the partitioning scheme adopted can result crucial for improving the overall computational performances.**

***Index Terms*—Cellular Automata, Parallelization, Sciddica, Partitioning.**

I. Introduction

Cellular Automata (CA) are parallel computing models [20],[21] that can be fruitfully applied for simulating natural phenomena, which can be modeled in terms of local interactions among their elementary parts. A distinctive characteristic of CA is the possibility of obtaining complex global behaviors by means of simple purely local rules. They are based on a regular tessellation of the space into a matrix of cells (called the cellular space). Each cell of the cellular space has a fixed shape and size, and embeds an identical finite automaton, whose input is given by the states of the neighbour cells. At time t=0, cells are in arbitrary states and the CA evolves step by step by changing the states of the cells at discrete time steps, by applying the same local rule of evolution, i.e. the cell's transition function, simultaneously (i.e. in parallel) to each cell of the CA. Input for the cell is given by the states of a predefined (usually small) set of neighboring cells, which is assumed invariant in space and time. Despite their simple definition, CA can exhibit very interesting complex global behavior. Moreover, from a computational point of view, they are equivalent to Turing Machines meaning that, in principle, they are capable to compute everything that can be computed (the Church-Turing thesis). Thanks also to this property – computational universality – CA gained a great consideration among the scientific community, have been employed to solve a great variety of different complex problems and have proven to be a valid alternative to differential equations in simulating complex natural phenomena [18].

Extended Cellular Automata [12] (XCA) represents an extension of the original CA computational paradigm. Many applications have proven that the approach behind XCA can make the modeling of some complex systems more straightforward. In the XCA framework, the cell's state is decomposed in substates, each of them representing the set of admissible values of a given characteristic assumed to be relevant for the modeled system and its evolution (e.g., lava temperature, lava thickness, etc, in the case of a lava flow model). Nevertheless, CA (and XCA) models can be straightforwardly implemented on parallel computers due to their underlying parallel nature [11],[8].

This paper presents the application of a parallelization strategy for CA, applied to a landslide computational model. In particular, two cellular space partitioning have been taken into account, namely a monodimensional and two-dimensional block scheme partitioning. Starting from previous theoretical studies presented in [14], where the parallel execution of CA was modeled by a Petri net, we here present results of a parallel execution of a real CA model for landslide simulation. Actually, considering that communication times were considered negligible in [14], in the present work communication times are estimated by a simple analytical study and evaluated for the specific case-study. The paper is organized as follows. In the next section we present the adopted parallelization schemes, together with estimation of the communication and

2377-5750/17 $31.00 © 2017 IEEE
DOI 10.1109/PDP.2017.84

synchronization burden. Section III reports the adopted CA model which was used as case-study, while section IV shows the results referred to the execution on a 16-node cluster. Eventually, section V reports discussion and future outcomes of the proposed research.

II. Parallelizing CA Execution through Space Partitioning

In many parallel and distributed computing frameworks, the approach of partitioning the space or territory and assigning each region to a specific computing node is a valuable and efficient strategy to improve the performances in terms of efficiency and scalability [1][17][5][6][4].

In a parallel execution context, a cellular automata can be partitioned by considering a mono-dimensional or a two-dimensional partitioning schema as depicted in Figure 1. The assignment of computation to the nodes follows the space partitioning: each partition of the territory, or "region", can be assigned to a different computing node, which is in charge of executing the transition rules of all the cells belonging to this specific region. The transition rule of a cell is evaluated on the basis of the states of its neighbour cells. Hence, to execute the transition rules of the cells located in the borders of a region, information must be received from the adjacent computing nodes. For this reason, each node must synchronize with the neighbour nodes.

The *visibility radius* – referred to as r in the following – delimits the boundary of the neighbourhood, i.e., the cells involved in computing a transition function of a cell. Figure 2 shows a scenario where the neighbourood of a cell falls into two different adjacent regions (if mono-dimensional space partitioning is adopted) or into four different adjacent regions (in the case of two-dimensional space partitioning).

Access to remote data must be *correct* and *efficient*. Correctness, in this context, means working always with updated information. For example, two computing nodes could work with different status of the same cell because they run different steps at the same time. To avoid such an incorrect access to data, a step-by-step duplication mechanism can be adopted, which consists in replicating the edge areas of adjacent regions. Such areas, referred to as *borders*, are kept aligned by exchanging at each step *update messages* between the adjacent computing nodes, i.e., the nodes that manage the corresponding adjacent regions. This data exchange ensures that data is always updated at the last step. Replication of borders data is a valuable solution also from the efficiency point of view, as it ensures that all transition function are computed using only the local data and the replicas of the neighbour border data without any need to engage network operations. This increases the efficiency of operations on data and helps to reduce inter-node communications, thus improving performances.

The border area of a region is composed of two distinct parts: the *local border* and the *mirror border*. Figure 3 and Figure 4 show the borders in the cases of mono-dimensional and two-dimensional space partitioning. The local border is managed by the local node and its content is replicated in the mirror border of the adjacent nodes. At each step, all the modifications occurred in a local border are gathered and transmitted to the adjacent nodes. For example, information about the updates occurred in the local border of Node 1 of Figure 3 are sent to Node 2, which applies the updates in its mirror border. Analogously, information in the mirror border of Node 1 is aligned with the updates occurred in the local border of Node 2. In the case of two-dimensional partitioning, see Figure 4, an update message may be sent to more than one adjacent nodes.

A. Communication burden

In this section, we analyze the amount of data that is involved in communications during the parallel execution of a CA using the approach described so far. We first analyze the case of mono-dimensional partitioning, then the case of two-dimensional partitioning, and then we provide a comparison of the two approaches when both options are available. Data must be exchanged to keep regions informed about the updates occurred in the adjacent regions, more specifically in the region borders, as illustrated in Section II. Therefore, the area of the borders are a good proxy variable[1] for the estimation of the communication burden. The scenario under consideration consists of a toroidal rectangular territory with horizontal size equal to L space units and vertical size equal to H cells. The CA is partitioned into N regions through a number of horizontal cuts, C_l, and a number of vertical cuts, C_h. Each region has horizontal size $l = L/C_h$ and vertical size $h = H/C_l$, and is assigned to an associated computational node for execution.

To analyze the case of mono-dimensional space partitioning we assume, without loss of generality, that the space is partitioned through vertical cuts. Let us consider the sum of the areas of the left and right borders of a single region, shown in dark grey in Figure 5. This quantity, denoted as B_l and measured in squared cells, is taken as a proxy variable for the communication burden in charge of a single node. For the sake of simplicity, we refer to this quantity simply as "communication burden", but it is implicitly taken into account that, to compute the actual amount of exchanged communication, the amount of data transmitted per cell must be considered. For the case of mono-dimensional space partitioning, the communication burden of a single node, denoted as B_l, is equal to:

$$B_l = 2hr = 2Hr \tag{1}$$

It is noticed that B_l depend on the vertical size H, while they do not depend on the horizontal size L. Therefore, to minimize the value of the communication burden, it is convenient to consider the shorter side of the territory as the vertical side, so that $H <= L$. Moreover, the communication

[1]For the sake of clearness, here we consider the meaning of the term "proxy" as it is used in statistics, i.e., as a variable that helps to estimate another variable which is more complex to compute.

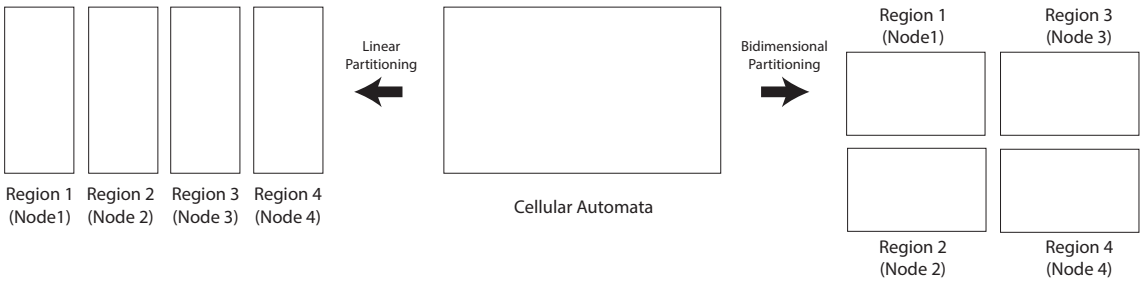

Fig. 1. The cellular space partitioned into regions which are associated with parallel computing nodes. Two alternative types of partitioning are shown, mono-dimensional and two-dimensional.

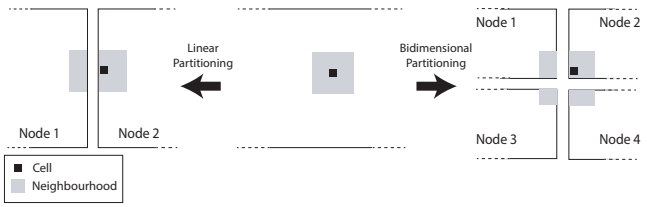

Fig. 2. A cell's neighbourhood overlapping more regions

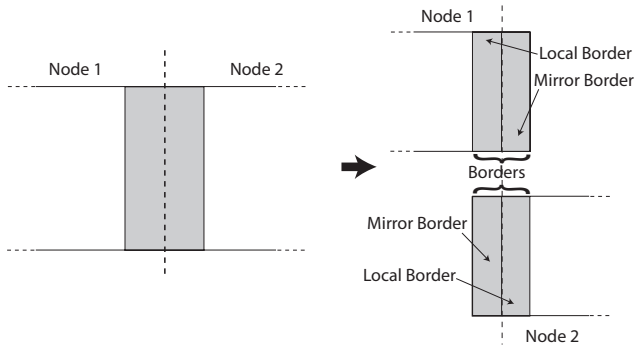

Fig. 3. Border areas of two adjacent nodes in the case of mono-dimensional partitioning

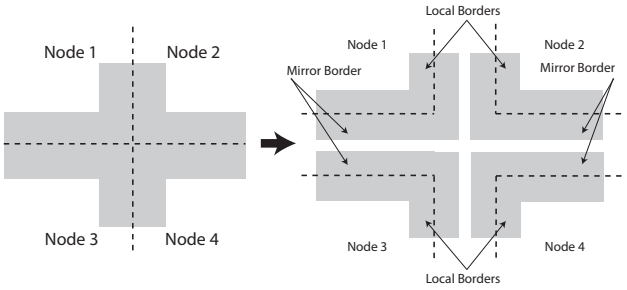

Fig. 4. Border areas of four adjacent nodes in the case of two-dimensional partitioning

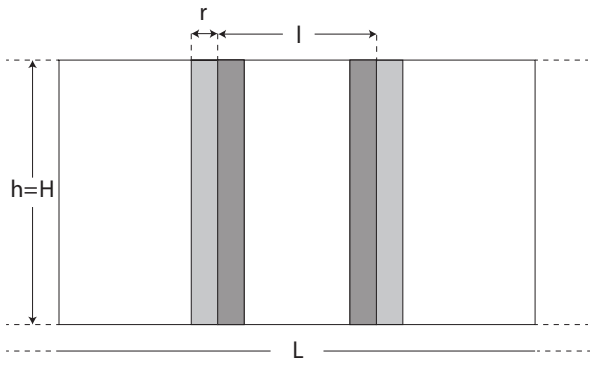

Fig. 5. Mono-dimensional space partitioning.

burden of a single node, B_l, does not depend on the number of involved computational nodes.

In the case of two-dimensional space partitioning, not all the borders are replicated on a single adjacent region, as in the mono-dimensional case: some borders must be replicated on two or more regions, depending on how the space is partitioned.

For example, Figure 6 pictures a two-dimensional space partitioning. The borders of the central region are distinguished and labeled with numbers "1" and "3", depending if the corresponding area is replicated on one or three adjacent regions, respectively. In this scenario, the communication burden should be computed by properly weighing the contributions of the two kinds of border area. Specifically, the sum of the four areas labeled with "3", denoted as A_3, is equal to A_3=$4r^2$. The sum of the areas labeled with "1", denoted as A_1, is equal to A_1=$2r(l+h)-8r^2$. When computing the communication burden, the quantity A_3 is multiplied by three, since the corresponding areas must be replicated on three regions. The communication burden of a single region is then given by:

$$B_b = A_1 + 3A_3 = 2(l+h)r + 4r^2$$

In the following, we focus on the impact that the shape of the regions (i.e., how *stretched* the rectangles are) has on the communication burden. In fact, for a given territory,

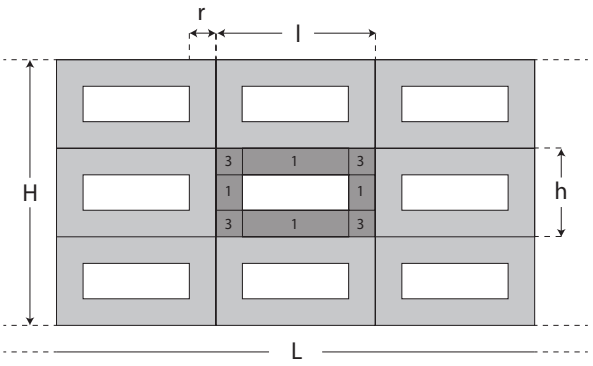

Fig. 6. Grid-like two-dimensional space partitioning.

while the area of a region only depends on the number of regions N, the communication burden B_b explicitly depends on the semiperimeter $(l+h)$, see expression (II-A), then on the regions shape. In the following it is shown that the minimum overhead is obtained when regions are square-shaped.

We express B_b in terms of l, by using the substitution $h = \frac{S}{l}$, where $S = l \cdot h$ is the area of the region. We obtain:

$$B_b(l) = 2(l + \frac{S}{l})r + 4r^2$$

We derive the expression and obtain:

$$\frac{dB_b(l)}{dl} = \frac{l^2 - S}{l^2} \cdot 2r$$

The minimum of $B_b(l)$ is obtained by setting the derivative equal to zero, which gives:
$l = \sqrt{S}$.

Since the second derivative at $l = \sqrt{S}$ is positive, the minimum communication burden is achieved when the region has a squared shape. This can be obtained when the number of vertical and horizontal partitions[2] of the CA are proportional to the respective sizes of the overall territory, i.e., C_l/C_h=H/L. Of course, the admissible options for the space partitioning also depend on the number of nodes N, since there is the constraint $N = C_l \cdot C_h$.

In several cases, the type of space partitioning, mono-dimensional or two-dimensional, is driven by constraints related to the application domain, e.g., the type of data that needs to be processed, the territorial distribution of sensors, etc. In other cases, however, there is more freedom to choose the more convenient territory partitioning. Therefore, it is interesting to compare the communication burdens corresponding to mono-dimensional and two-dimensional space partitioning, to see if one of the two options is more efficient than the other, and in which cases. Specifically, we do the comparison in a scenario where the sizes L and H are given. Moreover, in the case of two-dimensional partitioning we consider the case of square-shaped regions, since this proved to be the choice that minimizes the communication burden, as discussed before.

The two-dimensional partitioning has an equal or lower overhead when the ratio of expression (II-A) to expression (1) is equal or lower than one:

$$\frac{B_b}{B_l} = \frac{2r(l+h) + 4r^2}{2rH} <= 1 \qquad (2)$$

Using the equalities l=h and $C_l/C_h = H/L$, which correspond to square-shaped regions, and considering that the value of the operational radius r is typically much lower than the height of the entire territory H, the inequality (2) is solved for $C_l >= 2$ and $C_h >= 2 \cdot L/H$. It also corresponds to the inequality $N >= 4 \cdot L/H$. This means, for the case of square-shaped territory (i.e., L=H), that the two-dimensional partitioning is a better choice when the number of nodes is equal or larger than 4, and the convenience increases with larger numbers of nodes. If the territory is rectangular, the minimum number of nodes that makes the two-dimensional case more convenient increases as the territory is more and more stretched. For example, this minimum number is equal to N=16 if L=$4H$. Indeed, it is intuitive that the mono-dimensional partitioning becomes more efficient when the territory extends mostly on one dimension.

B. Synchronization burden

So far, the analysis has taken into account only the communication burden and how this aspect depends on the kind of space partitioning. Another important issue concerns the degree of synchronization of the whole system. Often, CA are parallelized using a typical master-slave approach [3] where a `master` node is in charge of coordinating the execution of a set of parallel `slave` nodes. For each step, the master node is notified by all the slaves nodes when they have completed the current step, afterwards the master node triggers the execution of the next step for all the slave nodes. With this approach, the execution is carried out using an all–to–all synchronization strategy, i.e., a node can start a new step only when all the nodes have finished the previous step. Our proposal concerns a fully distributed approach in which the nodes interact with each other in a peer-to-peer fashion without any master node. The distributed approach enables us to implement a "partial" synchronization in which each node needs to receive only the border replicas from its neighbour nodes to undertake the next step. Relaxing the all–to–all synchronization requirement has a positive effect on the overall performance, as shown in [14].

The synchronization burden is also related to the type of space partitioning. Indeed, with mono-dimensional partitioning, each node has up to 2 neighbour nodes, on the left and on the right, while with two-dimensional partitioning, each node has up to 8 neighbour nodes, see Figure 6 [3]. Therefore, the synchronization burden is larger with two-dimensional partitioning. This tends to counterbalance the advantage that the

[2]It may be useful to notice that, in a toroidal space, the number of horizontal partitions is equal to the number of vertical cuts, and vice versa.

[3]In a non-toroidal scenario, the regions located at the borders of the territory can have a lower number of neighbors.

two-dimensional partitioning has, with respect to the mono-dimensional partitioning, in terms of communication burden. The experimental evaluation, discussed in the next section, has shown that the impact of communication burden on the overall performance is higher than the impact of synchronization burden. This practical evidence allows us to conclude that two-dimensional space partitioning can be considered as the best strategy, at least in the case of CA requiring large computational resources.

III. The case study: the SCIDDICA CA model

Among different phenomena, flow-type landslides (e.g. debris flows) are profitably modeled by defining local interactions among their constituent parts. For this type of phenomenon, CA have been adopted by many authors with satisfactory results [16][7][13][15][19]. Among these efforts, the CA SCIDDICA family models were developed for simulating slow-moving earth flows (SCIDDICA release "T", [2]). Nevertheless, depending on its kinetic energy, a landslide can also show significant inertial properties (e.g., can move upward, along a slope, and override obstacles). Accordingly, the original model was modified: the "run-up" was added, and the case of study of the Mt. Ontake (Japan) debris avalanche was analysed (SCIDDICA release "O", [10]). After the May 1998 disaster in Campania (Italy), the model has further been extended (SCIDDICA family "S"), in order to consider some peculiar features of rapid debris flows. In these models, as the amount of material eroded along the path of a debris flow can usually be significant – greatly increasing the initial volume of the soil slip – the process of erosion of the regolith overlying the bedrock has been included into the model. Secondly, the original mechanism of distribution of the landslide debris among the central cell and the neighbouring ones has been improved. The need for simulating the process of "progressive" erosion of the regolith has led to further modifications and the release "S3" has thus been obtained.

For the SCIDDICA S3-hex version adopted in this work for the experiments, the CA neighbourhood considered is the hexagonal one, were each cell is adjacent to 6 (identical) neighbour ones. Three elementary processes constitute the transition function of the model, which are locally evaluated at each step of computation: (1) debris distribution among the cells, according to pressure gradients across the cells and to flow rheology; (2) run-up determination, used for evaluating the landslide ability of moving upslope; (3) mobilisation of the soil cover, i.e., erosion and transformation into landslide material, according to the energy of the flowing mass. At the beginning of each simulation, the substates of the cells are initialised by means of input matrices. In this way, the location and extent of each landslide source to be simulated are specified, together with the thickness of regolith. This latter overlies the bedrock and can be eroded by the flowing debris: it must then be considered for computing changes to the original volume of the landslide. Please refer to [9] for more details on the SCIDDICA S3-hex model.

IV. Experimental results

The strategies presented in Section II have been applied for the parallelization of the SCIDDICA model, in the scenario of the Sarno landslide occurred in 1998, by using a 767×925 CA space executed for 2500 computational steps. The experiments were carried out by varying the number of computing nodes of a cluster in which each node has an Intel Xeon E5-2670 CPU with 2.60GHz and 128GB RAM. The nodes are interconnected with an Intel Corporation I350 Gigabit Network. Each node hosts a portion of the CA space and execution is performed using the Java platform. Moreover, communication among nodes is achieved through standard TCP/IP connections using the Java socket technology. Figures 7 and 8 show the execution time and the speedup obtained by considering the different types of space partitioning – mono-dimensional and two-dimensional – and number of computing nodes. As predicted in Section II-A, the two-dimensional partitioning outperforms the mono-dimensional one starting from approximately 4 computing nodes[4], as can be noticed in both figures. As expected, the speedup improves with the increase of the number of computing nodes in both considered partitioning schemes, witnessing the good system scalability. In addition, also the advantage of two-dimensional partitioning with respect to mono-dimensional partitioning increases with the number of nodes.

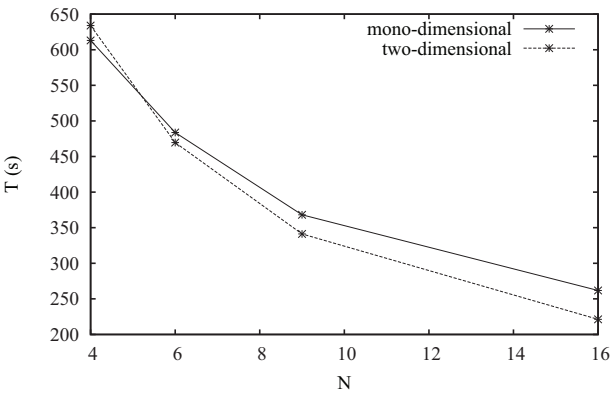

Fig. 7. Computational times vs. the number of computing nodes.

As a conclusive remark, it is worth noting that the two-dimensional space partitioning could be infeasible for some numbers of nodes, e.g., for a prime number of nodes. In other cases, although it is possible to achieve a two-dimensional partitioning, this can only be obtained with an "imbalanced" partitioning, i.e., with a number of partitions along one dimension which is much greater than in the other dimension. For example, with 22 nodes, the only possible two-dimensional partitioning is with 2x11 partitions. Such constraints on the partitioning, related to mere numerical issues, can limit or

[4]In Section II-A, we found that the two-dimensional partitioning is advantageous starting from 4 nodes, but this result was derived for the case of square-shaped territory, and without considering the synchronization burden. This is the reason why in the experiments the advantage of two-dimensional partitioning appears starting from a number of nodes slightly larger than 4.

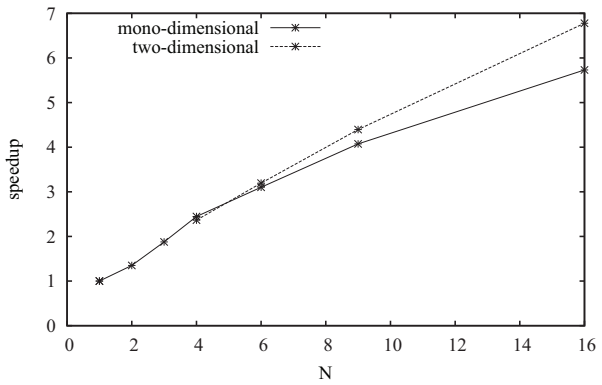

Fig. 8. Speedup vs. the number of computing nodes.

nullify the advantage of two-dimensional partitioning, even when a large number of nodes is employed. To cope with this issue, when a large number of nodes is available, one can consider to use only a subset of the available nodes that permits a more "squared" partitioning (e.g., if 74 nodes are available, an 8×8 partitioning can be the most efficient choice even if 10 nodes are not utilized).

V. Conclusions

In this paper we study different space partitioning schemes for parallelizing CA models. As known for distributed memory architectures, partitioning the CA space is a good solution for minimizing inter-node communication, achieving good performance and scalability. In our study, two main partitioning strategies are envisioned: the mono-dimensional and the two-dimensional ones. The two strategies are compared analytically by evaluating the burdens of communication, and are tested on the well-known SCIDDICA landslide model. In order to improve the overall performances, the parallel execution of the CA is carried out using a "partial" synchronization among the computing nodes instead of the all–to–all synchronization of the typical master-slave approach. The experiments show a good scalability for the system and suggest two-dimensional partitioning as the best choice when a large number of computer nodes are available.

Ongoing and future work are geared at the following:

- extending the experimental part by comparing our approach to the typical all–to–all synchronization;
- evaluating different approaches for parallelizing the CA execution also inside each region;
- integrating the approach with gpu- multi-gpu systems;
- studying the automatic parallelization of CA models when heterogeneous platforms for execution are available (e.g. by means of OpenCL).

References

[1] P. Albuquerque and A. Dupuis. A parallel cellular ant colony algorithm for clustering and sorting. In S. Bandini, B. Chopard, and M. Tomassini, editors, *Cellular Automata*, volume 2493 of *Lecture Notes in Computer Science*, pages 220–230. Springer Berlin Heidelberg, 2002.

[2] MV Avolio, Salvatore Di Gregorio, Franco Mantovani, Alessandro Pasuto, Rocco Rongo, Sandro Silvano, and William Spataro. Simulation of the 1992 tessina landslide by a cellular automata model and future hazard scenarios. *International Journal of Applied Earth Observation and Geoinformation*, 2(1):41–50, 2000.

[3] Mario Cannataro, Salvatore Di Gregorio, Rocco Rongo, William Spataro, Giandomenico Spezzano, and Domenico Talia. A parallel cellular automata environment on multicomputers for computational science. *Parallel Computing*, 21(5):803–823, 1995.

[4] Eugenio Cesario and Domenico Talia. Distributed data mining patterns and services: an architecture and experiments. *Concurrency and Computation: Practice and Experience*, 24(15):1751–1774, 2012.

[5] Franco Cicirelli, Agostino Forestiero, Andrea Giordano, and Carlo Mastroianni. Transparent and efficient parallelization of swarm algorithms. *ACM Trans. Auton. Adapt. Syst.*, 11(2):14:1–14:26, June 2016.

[6] Franco Cicirelli, Andrea Giordano, and Libero Nigro. Efficient environment management for distributed simulation of large-scale situated multi-agent systems. *Concurrency and Computation: Practice and Experience*, 27(3):610–632, 2015.

[7] Aldo Clerici and Susanna Perego. Simulation of the parma river blockage by the corniglio landslide (northern italy). *Geomorphology*, 33(1):1–23, 2000.

[8] Donato DAmbrosio and William Spataro. Parallel evolutionary modelling of geological processes. *Parallel Computing*, 33(3):186–212, 2007.

[9] S Di Gregorio, G Iovine, et al. Simulating debris flows through a hexagonal cellular automata model: Sciddica s3 hex. *Natural Hazards and Earth System Science*, 3(6):545–559, 2003.

[10] S Di Gregorio, F Nicoletta, R Rongo, M Sorriso-Valvo, G Spezzano, and D Talia. Landslide simulation by cellular automata in a parallel environment. In *Proceedings of 2nd International Workshop Massive Parallelism: Hardware, Software and Applications*, pages 392–407, 1994.

[11] Salvatore Di Gregorio, Giuseppe Filippone, William Spataro, and Giuseppe A Trunfio. Accelerating wildfire susceptibility mapping through gpgpu. *Journal of Parallel and Distributed Computing*, 73(8):1183–1194, 2013.

[12] Salvatore Di Gregorio and Roberto Serra. An empirical method for modelling and simulating some complex macroscopic phenomena by cellular automata. *Future generation computer systems*, 16(2):259–271, 1999.

[13] Giuseppe Filippone, Donato D'ambrosio, Davide Marocco, and William Spataro. Morphological coevolution for fluid dynamical-related risk mitigation. *ACM Trans. Model. Comput. Simul.*, 26(3):18:1–18:26, January 2016.

[14] Gianluigi Folino, Andrea Giordano, and Carlo Mastroianni. Scalable asynchronous execution of cellular automata. In *Proceedings of the 2nd International Conference Numerical Computations: Theory and Algorithms (NUMTA–2016)*, volume 1776.

[15] Federica Luc, Donato DAmbrosio, Gaetano Robustelli, Rocco Rongo, and William Spataro. Integrating geomorphology, statistic and numerical simulations for landslide invasion hazard scenarios mapping: An example in the sorrento peninsula (italy). *Computers & Geosciences*, 67:163 – 172, 2014.

[16] Bruce D Malamud and Donald L Turcotte. Cellular-automata models applied to natural hazards. *Computing in Science & Engineering*, 2(3):42–51, 2000.

[17] Carlo Mastroianni, Eugenio Cesario, and Andrea Giordano. Balancing speedup and accuracy into smart city parallel applications. In *Proc. of Euro-Par Workshops 2016*, Grenoble, France, August 2016. Best Paper Award.

[18] Tommaso Toffoli. Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics. *Physica D: Nonlinear Phenomena*, 10(1):117–127, 1984.

[19] Giuseppe A Trunfio, Donato DAmbrosio, Rocco Rongo, William Spataro, and Salvatore Di Gregorio. A new algorithm for simulating wildfire spread through cellular automata. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 22(1):6, 2011.

[20] John Von Neumann, Arthur W Burks, et al. Theory of self-reproducing automata. *IEEE Transactions on Neural Networks*, 5(1):3–14, 1966.

[21] Stephen Wolfram. *A new kind of science*, volume 5. Wolfram media Champaign, 2002.