Cellular Automata + Parallel Computing = Computational Simulation

Domenico Talia
ISI-CNR
c/o DEIS, Università della Calabria, 87036 Rende, Italy
e-mail: talia@si.deis.unical.it

Keywords: cellular automata, parallel programming, computational science

ABSTRACT

In the latest years a novel method has been added in science to theory and laboratory experiments for studying and solving scientific problems. That method can be defined as *computational simulation*. Computational simulation is based on the use of computers for modeling and simulation of complex systems in science and engineering. According to this approach, a computer equipped with problem-solving software tools may represent a *virtual laboratory* where it is possible to build a model for a given problem and run it under different conditions. Parallel computers represent a class of computing architectures that might effectively support the computational simulation approach. In particular, parallel computers are well suited for implementing inherently parallel computing abstract models such as cellular automata [1]. This paper describes how the combination of the cellular automata model with parallel computing techniques and systems can be very fruitful in supporting the computational simulation approach in science and engineering

INTRODUCTION

Formal theory definition and experiments are the basic methods for studying and verifying scientific discoveries. In the latest years a third method has been added in science to theory and laboratory experiments for studying and solving scientific problems. That third approach can be defined as *computational simulation*. Computational simulation is based on the use of computers for modeling and simulation of complex phenomena and systems in science and engineering. According to this approach, a computer equipped with problem-solving software tools may represent a *virtual laboratory* where it is possible to build a model for a given problem and run it under different conditions.

Parallel computers represent a class of architectures that might effectively support the computational simulation approach. They exploit the computing power of a large number of processors running in parallel by providing high performance in supporting efficient execution of complex scientific applications. In particular, parallel computers are well suited for implementing inherently parallel computing abstract models such as cellular automata, neural networks, and genetic algorithms that represent significant mathematical models for describing complex scientific phenomena.

In this paper we discuss the combined use of parallel computing and cellular automata in computational simulation and show how by developing programmable cellular automata on parallel computers it possible to provide computational simulation environments to be used for solving real problems in science and engineering. As an example of this approach we present the main features of cellular automata parallel software environments and how those features allow to solve real world problems.

Cellular automata (CA) are very effective in solving scientific problems because they can capture the essential features of systems in which the global behavior arises from the collective effect of large numbers of locally interacting simple components [1]. A cellular automaton is a discrete dynamic system composed of a set of cells in a one-dimensional or multi-dimensional lattice. The global behavior of the system is determined by the evolution of the states of all cells as a result of multiple interactions.

Although the cellular automata theory has been defined several decades ago, only in recent years it became significant as a tool for modeling and simulation of complex systems. This occurred thanks to the implementation of cellular automata on massively parallel computers. These new architectures made possible designing and development of high-performance software environments based on the cellular automata theory. Significant examples of these environments are CAM [2], CAMEL [3], StarLogo [4], CAPE [5], and NEMO [6]. These cellular automata parallel environments allow the exploitation of the inherent parallelism of the cellular automata model for the efficient simulation of complex systems that can be modeled by a very large number of simple elements (cells) with local interaction only.

By means of the systems mentioned before and by other CA systems, cellular automata have been used to solve complex problems in many fields of science, engineering, computer science, and economy. In particular, parallel cellular automata models are successfully used in fluid dynamics, molecular dynamics, biology, genetic, chemistry, road traffic flow, cryptography, image processing, environment modeling, and finance. This paper describes in detail how the marriage of the cellular automata theory with parallel computing is very fruitful in supporting the computational simulation approach in science and engineering. Practical examples are discussed by means of real parallel CA systems such as CAMEL, CAM, CAPE, and Starlogo.

CELLULAR AUTOMATA

A cellular automaton (CA) can be defined as a d-dimensional Euclidean space, partitioned into cells of uniform size, each one embedding an identical elementary automaton (ea). Input for each ea is given by the states of the elementary automata in the neighbouring cells, where neighbourhood conditions are determined by a pattern invariant in time and constant over the cells. Figure 1 shows a von Neumann neighbourhood, a very simple and classic neighbourhood, in a two dimensional lattice.

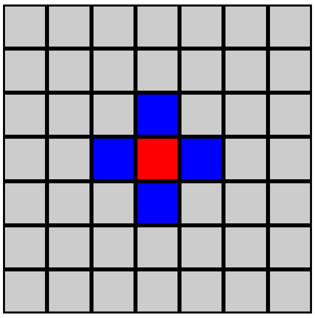

Fig. 1. A two dimensional CA lattice with a von Neumann neighbourhood. The cell above and below, right and left from each cell compose the von Neumann neighbourhood.

At the time t=0, ea are in arbitrary states and the CA evolves changing the state of all ea at discrete times, according to a local rule. Each cell in the regular spatial lattice can have any one of a finite number of states. As mentioned before, the states of the cells in the lattice are updated according to a local rule called *state transition function*. That is, the state of a cell at a given time depends only on its own state in the previous time step and the states of its nearby neighbors at the previous time step [7]. All cells of the automaton are updated synchronously in parallel. Thus the state of the entire automaton advances in discrete time steps. Therefore, the transition function plays in cellular automata a role analogous to that of the evolution equation in classical dynamical models. Therefore, a cellular automaton is a discrete dynamic system and the state of entire automaton

advances in discrete time step. The global behavior of the system is not directly specified but it is determined by the evolution of the states of all cells as a result of multiple interactions.
Cellular automata are a powerful tool for modeling natural phenomena and were one of the first parallel computing abstract models. Conceived by John Neumann in the 1950's to investigate self-reproduction, CA have been mainly used for studying parallel computing methods and the formal properties of complex systems models. However, with the rapid advances in computational resources during the 1980's, CA have become more and more used for computer simulation and modeling in several application areas. Recently, it has been shown that CA models can be effectively used both as a realistic approach to define abstract parallel machines and as a programming methodology for computational science on parallel computers. CA capture the peculiar features of systems which may be seen to evolve exclusively according to local interactions of their constituent parts, and guarantee computational universality. Furthermore, applied aspects of modeling have been widely investigated from a theoretical viewpoint.

PARALLEL CA SYSTEMS

Without the use of high-performance parallel computers the CA model would be of no practical use for solving real world problems. There are two possible alternatives which allow to achieve high performance in the implementation of CA. The first one is the design of special hardware devoted to the execution of CA. The second alternative is based on the use of commercially-available parallel computers where the state of cells can be updated simultaneously.
CAM (Cellular Automata Machine) is the most significant example of a specialized hardware which has been designed to run CA simulations. Although the CAM offers a high-level environment for programming CA and can run CA simulations in an efficient way, it is limited in the size of the automata which can be simulated and in the number of states per cell. Furthermore, it is a specialized machine which cannot be utilized as a general purpose computer.
On the other hand , highly parallel computers offer the most natural architecture for a CA machine. These systems are based on a number of interconnected processing element (PE) which perform a task concurrently. Both SIMD and MIMD architectures are suitable to support CA implementation achieving high parallel efficiency. We prefer to use the MIMD model, as this allows us to deal with various irregularities on a microscopic level of the algorithm asynchronously and to efficiently simulate also heterogeneous systems. However, on a higher level of a abstraction we will synchronize the parallel component explicitly, as this is the only way to maintain global invariance.
According to the second approach in the latest years several parallel cellular software environments have been developed. The main features shared by parallel cellular software environments are

- a high-level programming layer for designing software models of complex phenomena which is independent from the underlying parallel architecture;
- a graphical user interface (GUI) that allows the visualization of the complete evolution of a simulation of dynamic systems and the display of numerical values associate to a simulation;
- monitoring and tuning facilities at run-time that allow to change the simulation parameters according to the model evolution and eventually to the expected values;
- scalable high-performance that allows these environments to support the efficient execution of very complex real world phenomena that were impossible to simulate on sequential computers.

Significant examples of these parallel cellular environments are CAMEL [3], StarLogo [4], NEMO [6], and CAPE [5]. These cellular automata parallel environments allow the exploitation of the inherent parallelism of the cellular automata for supporting the efficient simulation of complex systems that can be modeled by a very large number of simple elements (cells) with local interaction only. Due to constrain of the paper length we give a very short outline of these systems. For each of them, in the next section we discuss a significant application in the computational simulation area.

The CAMEL environment has been implemented on a message-passing MIMD parallel computer. It has three main components, the run-time system, the user interface and the visualization system. The CAMEL environment implements a cellular automaton as a SPMD (Single Program, Multiple Data) program. In fact, CA are implemented as a number of processes each one mapped on a distinct PE that executes the same code on different data. According to this approach a user must specify only the transition function of a single cell of the system he wants to simulate by CARPET [8], a high-level cellular language defined as an extention of the C language.

StarLogo is a CA-based extension of the Logo programming language which has been designed by Mitchel Resnick at MIT especially for educational use. Starlogo provides simple constructs to define the evolution rules of the cells that compose a cellular automaton and it allows a user to observe the global behavior of all cells on a graphical screen. A Starlogo environment has been implemented on the Connection Machine.

CAPE (Cellular Automaton Programming Environment) is a lattice gas parallel cellular system implemented at EPCC on a Meiko CS-1. The programming code can be written in FORTRAN. In fact the CAPE design is centered on a quick export of sequential FORTRAN programs from sequential to parallel machine. CAPE provides also graphics facilities for the visualization of the CA program results.

Finally, NEMO (Neighbourhood Modelling) is an environment based on cellular automata especially designed for raster processing. NEMO has been implemented on a MIMD distributed memory computer providing, like the other CA systems mentioned here, transparent parallelism by hiding all parallel details coming from the architecture. Moreover, NEMO provides a display facility for the visualization of graphical data from the simulation.

APPLICATIONS IN SCIENCE AND ENGINEERING

In this section we present, for three of the systems mentioned above, a significant application in the computational simulation area.

A CAMEL Application

CAMEL has been used to simulate several real world phenomena such as the flow of lava from a volcano, the traffic flow in a freeway, and the evolution of a landslide have been simulated with good accuracy and achieving high performance [9]. Recently, CAMEL has been used in the CABOTO (Cellular Atomata for the Bioremoval of Toxic Contaminants) project within the PCI ESPRIT programme. The CABOTO main objective concerned the use of CA to model and simulate the bioremediation of contamined soils. In particular, such models describe the decontamination processes that can be performed by proper stimulation of the growth of indigenous bacteria. These models allow the prediction of real scale bioremediation operation from the knowledge of geological, chemical, and microbiological data and from results of suitable experiments performed in the laboratory or in a pilot plant.

The cellular automata model that has been used to simulate the phenomenon is composed by a large number of cells, where each cell being of "mesoscopic" size does not describe the phenomena which take place inside a single pore (typical pore dimensions being 10-100 μm) but those inside a portion of soil. Our model is three-dimensional and it is designed for real scale simulations in order to describe heterogeneous regions. The neighbourhood of each cell is composed by six other cells, four of them being at the same height (*north, south, east, west*), while the remaining two neighbouring cells are in the *up* and *down* positions. The model has a "layered" structure, where the first layer describes fluid flow through the soil, the second layer describes the behaviour of chemicals (either solutes or adsorbed on the pore surface) and the third layer describes the interaction between chemicals and biomass [10].

The performance results show that to simulate the full event of bioremediation (which takes 64 real days) on a 128x15x11 grid (21120 cells), are necessary about 19 simulation days using a single processor. Whereas only about 16 simulation hours are necessary using 32 processors. This important result shows the need of parallel computation in solving complex problems such as the bioremediation phenomenon, and the great efficacy of CA parallel implementation in exploiting parallelism.

A Starlogo Application

This Starlogo application simulates the spread of a fire through a forest. The fire starts on specific point of the forest, and spreads to neighboring trees. The fire spreads in four directions: north, east, south, and west. The fire's chance of diffusing towards all the forest depends critically on the density of trees in the forest. The forest fire problem is a typical example of problems studied in the percolation theory that analyses interactions between substances with different properties. Similar problems are the oil diffusion through a porous rock or the motion of electrons in superconductors composed of superconducting material and resistive material.
This Starlogo program allows to change the density of trees and repeats the simulation running to verify different evolutions of the fire spreading. A user can follow the evolution of the fire spreading on the computer monitor where the application is displayed as in figure 2.

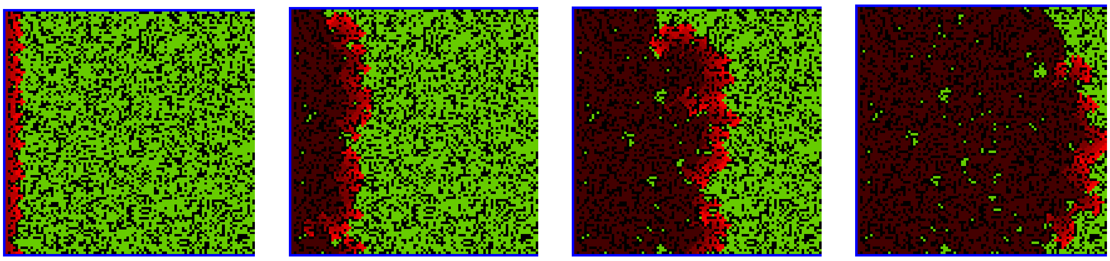

Fig. 2. Four snapshots of the forest fire simulation by Starlogo, the fire starts on the left edge of the forest, and spreads to neighboring trees.

A NEMO Application

The NEMO parallel system has been used by the PARADIGM group at Carleton University for developing computational application in the area of earthquake modeling, ice tracking, forest fire modeling, and response time modeling [6].
Here we briefly describes earthquake simulation by the NEMO system. As people know, major earthquakes are one of the most damaging natural disasters to human society. Studies of earthquakes in the past have helped provide insight into the behaviour of earthquakes. A well known power law (i.e., the Gutenberg-Richter b value) advanced the understanding and modeling of earthquake fractures. Laboratory experiments on various rock samples provide additional insight. Using the cellular automata model of NEMO, the model of Henderson et. al. has been implemented by Z. Wang to simulate the evolution of sample failure using fracture mechanism theory, which mimics the process of seismogenesis.
The model begins by initializing a raster with a fractal distribution of fracture toughness. Each cell is given a stress intensity, and a boolean value of either *solid*, when the local stress intensity is less than fracture toughness, or *broken* otherwise. The neighbourhood function applied to each cell of the raster causes a cell to increase or decrease its fracture toughness to reflect stress intensity of the cells in its neighbourhood . The simulation stops once no element has a local stress intensity greater

than its fracture toughness. Then, external stress is incremented and the above procedure iterates until a single crack across the raster (i.e., a major earthquake) occurs.

CONCLUSIONS

This paper discussed how the combination of the cellular automata model with parallel computing techniques and systems can be very fruitful in supporting the computational simulation approach in science and engineering. Some general-purpose parallel cellular automata systems have been presented and a few real world complex applications implemented on them have been discussed.
According to our research experience and the work of other research teams, we found that cellular automata represent an efficient paradigm for the computer solution of problems in science and engineering. Moreover, as stated also by Flynn [11], the cellular automata model allows to effectively use parallel computers achieving scalable performance. In summary, parallel cellular processing provide

- a new mathematical way to represent problems,
- the scalable performance of cellular MIMD processors, and
- an environment for an interdisciplinary effort for representing and simulating scientific and engineering problems.

All that we experienced in the latest years working on the implementation and use of a parallel cellular environment as a virtual laboratory for computational simulation.

REFERENCES

[1] J. von Neumann, *Theory of Self Reproducing Automata*, University of Illinois Press (1966).

[2] T. Toffoli and N. Margolus, *Cellular Automata Machines A New Environment for Modeling*, The MIT Press (1986).

[3] G. Spezzano G., D. Talia, S. Di Gregorio, R. Rongo, W. Spataro*, A Parallel Cellular Tool for Interactive Modeling and Simulation*, IEEE Computational Science & Engineering 3, p. 33 (1996).

[4] M. Resnick, *Turtles, Termites, and Traffic Jams*, The MIT Press, Cambridge (1994).

[5] M.G. Norman, J.R. Henderson, G. Main, D. J. Wallace, *The Use of the CAPE Environment in the Simulation of Rock Fracturing*, Concurrency: Practice and Experience 3, pp. 687 (1991).

[6] D. Hutchinson, L. Kuttner, M. Lanthier, A. Maheshwari, D. Nussbaum, D. Roytenberg, J.-R. Sack, *Parallel Neighbourhood Modeling: Research Summary*, Proc. SPAA '96, Padua, Italy, pp. 204 (1996).

[7] S. Wolfram, *Computation Theory of Cellular Automata*, Comm. Math. Phys. 96, p. 15 (1984).

[8] Spezzano G. and Talia D.*, A High-Level Cellular Programming Model for Massively Parallel Processing*, Proc. 2nd Int. Workshop on High-Level Programming Models and Supportive Environments (HIPS97), IEEE Computer Society Press (1997).

[9] Cannataro, S. Di Gregorio, R. Rongo, W. Spataro, G. Spezzano, D. Talia, *A Parallel Cellular Automata Environment on Multicomputers for Computational Science*, Parallel Computing 21, pp. 803 (1995).

[10] S. Di Gregorio, R. Rongo, W. Spataro, G. Spezzano, D. Talia, *A Parallel Cellular Simulator for Bioremediation of Contaminated Soils*, in Development and Applications of Computer Techniques to Environmental Studies VI, Computational Mechanics Publ., C.A. Brebbia et al. (Eds.), pp. 685 (1996).

[11] M. J. Flynn, *Parallel Processors Were the Future and May Yet Be*, IEEE Computer 29, pp. 152 (1996).