Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 23 (2007) 671–679
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

An autonomic tool for building self-organizing Grid-enabled applications

Gianluigi Folino, Giandomenico Spezzano*

Institute of High-Performance Computing and Networking (ICAR), National Research Council (CNR), Via Pietro Bucci 41C, I-87036 Rende (CS), Italy

Received 15 December 2005; received in revised form 22 November 2006; accepted 24 November 2006
Available online 10 January 2007

Abstract

In this paper we present CAMELotGrid, a tool to manage Grid computations of Cellular Automata that support the efficient simulation of complex systems modeled by a very large number of simple elements (cells) with local interaction only. The study of these systems has generated great interest over the years because of their ability to generate a rich spectrum of very complex patterns of behavior out of sets of relatively simple underlying rules. Moreover, they appear to capture many essential features of complex self-organizing cooperative behavior observed in real systems. The middleware architecture of CAMELotGrid is designed according to an autonomic approach on top of the existing Grid middleware and supports dynamic performance adaptation of the cellular application without any user intervention. The user must only specify, by global criteria, the high level policies and submit the application for execution over the Grid.

Keywords: Grid computing; Autonomic computing; Cellular automata

1. Introduction

The emergence of Computational Grids [12] as the next generation distributed computing platform has enabled a new generation of applications based on seamless access, aggregation and interaction. For example, it is possible to conceive a new generation of scientific and engineering simulations of complex physical phenomena that combine computations, experiments, observations, and real-time data and can provide important insights into complex systems such as road traffic, image processing, landslide simulation and science of materials.

Many of these phenomena have been successful modeled and simulated by cellular automata (CA) [24]. CA are mathematical models for complex natural systems containing large numbers of simple identical components with local interactions. They consist of a lattice of sites, each with a finite set of possible values. The value of the sites evolve synchronously in discrete time steps according to identical rules. The value of a particular site is determined by the previous values of a neighborhood of sites around it. CA are discrete dynamical systems with simple construction but complex *self-organizing* behavior. Current CA packages are either specialized single node programs or they are programming environments for building parallel applications. Today, with the current trend for larger scale CA problem solutions, we need Grid-enabled implementations of CA.

Computational Grids provide the software and networking infrastructure to harness a heterogeneous environment that includes geographically distributed computer domains, to form a massive computing environment through which large scale problems can be solved. To achieve this goal, Grids need to support various tools and technologies that can guarantee security, uniform access, resource management, scheduling, application composition, computational economy and accounting [3].

Realizing software systems on the Grid requires not only the knowledge of standards such as OGSA (Open Grid Services Architecture) and tools like Globus [11], but also sophisticated paradigms that effectively hide the complexity of creating and deploying truly parallel Grid applications in the presence of dynamicity, adaptivity and fault tolerance.

High level problem solving environments (PSEs) provide a general, uniform framework allowing researchers to concentrate on their specific system of interest without being involved in the lower level parallelization tasks. However, building PSEs in a computational grid infrastructure [26,19] is a challenging task because the concurrent program, which

* Corresponding author.
E-mail address: spezzano@icar.cnr.it (G. Spezzano).

0167-739X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2006.11.003

represents the runtime of the application, must dynamically adapt to changing resource availability in the grid environment. Supporting dynamically configurable programs requires a programming paradigm and management techniques that deal with complexity, heterogeneity and uncertainty. This has led researchers to consider alternative programming paradigms based on the strategies used by biological systems that exhibit a self-organizing behavior and that have been recognized suitable for managing distributed resources.

A significant bio-inspired paradigm has been defined by IBM in its "Autonomic Computing" program [16]. An autonomic computing system is a system which has the capabilities of being self-defining, self-healing, self-configuring, self optimizing, etc. and is able to manage itself without involving the user, in the same way the autonomic nervous system regulates the body systems without conscious input from the individual. The user must only specify, by global criteria, the high level policies (runtime partitioning strategies, etc.) and submit the application for execution over the Grid.

In this paper, we present CAMELotGrid, the Grid-enabled version of the CAMELot (*Cellular Automata environMent for systEms ModelLing open technology*) PSE developed in the Esprit project COLOMBO [18,8]. CAMELotGrid is a new middleware designed on top of the existing Grid middleware, which uses autonomic Grid functionality to intelligently manage problem partitioning, problem piece deployment, runtime management, dynamic level of parallelism, dynamic load balancing, and, in future, even fault tolerance and recovery.

The remainder of this paper is organized as follows. Section 2 briefly presents an overview of CAMELot. Sections 3 and 4 describe how to specify autonomic requirements of a cellular application and the middleware architecture of CAMELotGrid. Section 5 illustrates the performance model for predicting application execution time and in Section 6 we evaluate the application performance using a landslide cellular model. Section 7 concludes with a summary.

2. CAMELot overview

CAMELot is a high performance simulation environment based on the CA formalism [25]. In our approach, a cellular algorithm is composed of all the transition functions of the cells that compose the lattice. Each transition function generally uses the same local rule, but it is possible to define some cells with different transition functions (*heterogeneous* cellular automata). Unlike early cellular approaches, in which cell state is defined as a single bit or a set of bits, we define the state of a cell as a set of typed sub-states. This allows extending the range of applications that can be programmed by cellular algorithms. Furthermore, we introduce a *logic* neighborhood that may represent a wide range of different neighborhoods inside the same radius and that may also be time-dependent. We have also implemented some mechanisms to observe and control the evolution of the automaton. The CAMELot simulation environment consists of:

- a graphic user interface (GUI) for editing, compiling, configuring, executing, visualizing and steering the computation. The GUI allows, by menu pops, to define the size of the CA, the number of the processors on which the automaton must be executed, and to choose the colors to be assigned to the cell sub-states to support the graphical visualization of their values;
- a software library to integrate raster GIS images into the CA. The raster information can consist of different variables such as altimetry, soil, temperature, vegetation, etc. In CAMELot these variables are associated with the sub-states where the transition function provides a dynamic alteration of the information. For instance, the temperature values can be changed by a simple model that updates the temperature with regard to the hour of the day;
- a load balancing algorithm similar to the *scatter decomposition* technique to evenly distribute the computation among processors of the parallel machine;
- a language, called CARPET [7], which can be used to define cellular algorithms and to perform steering commands when complex space and time events are detected.

CARPET is a language to program cellular algorithms and contains constructs to extend the range of interaction among the cells, introducing the concept of *region*, and to define algorithms to perform *computational steering*. It is a high-level language based on C with additional constructs to describe the rule of the state transition function of a single cell of a cellular automaton and to steer the application. A CARPET program is composed of a *declaration* part that appears only once in the program and must precede any statement, a *body* program that implements the transition function, and a *steering* part that contains a set of commands to extract and analyze system information and to perform steering.

Fig. 1 shows an example of application of these constructs. Two 3D regions are defined in a three-dimensional cellular automaton. The event expression checks whether the maximum and the minimum of the rainfall sub-state in a *region(zone*1*)* are equal. In case they are, the computation is stopped. If the sum of the rainfall values in another *region(zone*2*)* is greater than a threshold, then the value of the alpha parameter is changed. In any case, the computation is stopped after 10 000 generations.

3. CAMELotGrid: An autonomic PSE

CAMELotGrid [8,6] is a PSE that provides a complete integrated computing environment for CA programming, permits one to specify the global criteria defining the autonomic requirements of the application and to support the execution of cellular applications over the Grid. It extends the original CAMELot architecture for incorporating the features of self-configuring, self-optimizing, self-healing, etc., of an autonomic system, in order to realize a Grid-enabled middleware architecture where the runtime autonomic management of the application is done without any user intervention.

In CAMELotGrid a spatio-temporal problem can be modeled by a 2D or 3D array of cells where each cell represents a portion of a landscape. By CARPET a user can describe, using the *declaration*, *body* and *steering* part, the transition function that represents the cellular program of the