Cellular Automata and GPGPU: An Application to Lava Flow Modeling

Donato D’Ambrosio, University of Calabria, Italy
Giuseppe Filippone, University of Calabria, Italy
Rocco Rongo, University of Calabria, Italy
William Spataro, University of Calabria, Italy
Giuseppe A. Trunfio, University of Sassari, Italy

ABSTRACT

This paper presents an efficient implementation of the SCIARA Cellular Automata computational model for simulating lava flows using the Compute Unified Device Architecture (CUDA) interface developed by NVIDIA and carried out on Graphical Processing Units (GPU). GPUs are specifically designated for efficiently processing graphic data sets. However, they are also recently being exploited for achieving excellent computational results for applications non-directly connected with Computer Graphics. The authors show an implementation of SCIARA and present results referred to a Tesla GPU computing processor, a NVIDIA device specifically designed for High Performance Computing, and a GeForce GT 330M commodity graphic card. Their carried out experiments show that significant performance improvements are achieved, over a factor of 100, depending on the problem size and type of performed memory optimization. Experiments have confirmed the effectiveness and validity of adopting graphics hardware as an alternative to expensive hardware solutions, such as cluster or multi-core machines, for the implementation of Cellular Automata models.

Keywords: Cellular Automata, Computer Graphics, Computer Unified Device Architecture (CUDA), GPU Programming, Graphic Processing Units (GPU), Lava Flow Simulation

INTRODUCTION

Nowadays, parallel computing is seen as a cost-effective method for the fast and efficient resolution of computationally large and data-intensive problems (Grama et al., 2003). The great expansion of High Performance Computing (HPC) in different scientific and engineering fields has permitted the use of numerical simulations as a tool for solving complex equation systems which rule the dynamics of complex real phenomena, through which researchers can study the modelling of, for instance, a lava flow, fire spreading or traffic simulation. Usually, the modeler has to implement proper optimization strategies and when possible, parallelize the program. The type of parallelization needed in this latter phase depends on the kind of available parallel architecture. For instance, in the case of a distributed memory machine (such

DOI: 10.4018/jghpc.2012070102
as Beowulf clusters), this can be accomplished by means of MPI - Message Passing Interface (Snir et al., 1995). On the contrary, in the case of a multicore architecture (such as Intel’s Core i7 processor), a shared-memory or multithread implementation based on OpenMP (Chapman et al., 2007) can result in a better and more efficient solution. In recent years however, parallel computing has undergone a significant revolution with the introduction of GPGPU technology (General-Purpose computing on Graphics Processing Units), a technique that uses the graphics card processor (the GPU) for purposes other than graphics. Currently, GPUs outperform CPUs on floating point performance and memory bandwidth, both by a factor of roughly 100. As a confirmation of the increasing trend in the power of GPUs, leading companies such as Intel have already integrated GPUs into their latest products to better exploit the capabilities of their devices, such as in some releases of the Core i5 and Core i7 processing units. Although the incredible processing power of graphic processors may be used for general purpose computations, a GPU may not be suitable for every computational problem: only a parallel program that results optimized for GPU architectures can fully take advantage of the performance of GPUs. In fact, the performance of a GPGPU program that does not sufficiently exploit a GPU’s capabilities can often be worse than that of a simple sequential one running on a CPU, such as when data transfer from main memory to video memory results crucial. Nevertheless, GPU applications to the important field of Computational Fluid Dynamics (CFD) are increasing both for quantity and quality among the Scientific Community (e.g., Tolke & Krafczyka, 2008; Zuo et al., 2010).

Among the different methodologies used for modelling geological processes, such as numerical analysis, high order difference approximations and finite differences, Cellular Automata (CA) (von Neumann, 1966) have proven to be particularly suitable when the behaviour of the system to be modelled can be described in terms of local interactions. Originally introduced by von Neumann in the 1950s to study self-reproduction issues, CA are discrete computational models widely utilized for modeling and simulating complex systems. Well known examples are the Lattice Gas Automata and Lattice Boltzmann methods (Succi, 2004) which are particularly suitable for modelling fluid dynamics at a microscopic level of description. However, many complex phenomena (e.g., landslides or lava flows) are difficult to be modeled at such scale, as they generally evolve on large areas, thus needing a macroscopic level of description. Moreover, since it may be difficult to model these phenomena through standard approaches such as differential equations (cf. McBirney & Murase, 1984, for the case of lava flows), Macroscopic Cellular Automata (MCA) (Di Gregorio and Serra, 1999) can represent a valid alternative. Several successful attempts have been carried out regarding solutions for parallelizing MCA simulation models. Among others, CAMELot (Di Gregorio et al., 1997) and libAuToti (Spongola et al., 2008) represent valid solutions for implementing and automatically parallelizing MCA models on distributed memory machines while for shared memory architectures, some effective OpenMP parallelizations have been implemented for CA-like models, such as for fire spread simulation (Innocenti et al., 2009), Lattice Boltzmann models (Heuveline et al., 2009) or lava flow modelling (Oliverio et al., 2011). However, only some examples of GPGPU applications for CA-like models do exist (e.g., Fan et al., 2004; Tolke, 2008) and to our knowledge, none regarding the MCA approach. This paper presents a first implementation, in GPGPU environments, of a well-known, reliable and efficient MCA model widely adopted for lava flow risk assessment (e.g., Crisci et al., 2010), namely the SCIARA lava flow model. Tests performed on two types of GPU hardware, a GeForce GT 330M low cost commodity graphic card and a Tesla C1060 computing processor, have shown the validity of this kind of approach. In the following sections, after a brief description of the basic version of the SCIARA MCA model for lava flows, an overview of GPGPU paradigm together with the CUDA framework
Extending Dynamic Scheduling Policies in WorkflowSim by Using Variance based Approach
[www.igi-global.com/article/extending-dynamic-scheduling-policies-in-workflowsim-by-using-variance-based-approach/153971?camid=4v1a](www.igi-global.com/article/extending-dynamic-scheduling-policies-in-workflowsim-by-using-variance-based-approach/153971?camid=4v1a)

Facilitating Biodefense Research with Mobile-Cloud Computing
[www.igi-global.com/chapter/facilitating-biodefense-research-mobile-cloud/64523?camid=4v1a](www.igi-global.com/chapter/facilitating-biodefense-research-mobile-cloud/64523?camid=4v1a)