J. Parallel Distrib. Comput. 121 (2018) 53–70

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

The Open Computing Abstraction Layer for Parallel Complex Systems Modeling on Many-Core Systems

Donato D'Ambrosio [a,*], Alessio De Rango [a], Marco Oliverio [b], Davide Spataro [c], William Spataro [a], Rocco Rongo [a], Giuseppe Mendicino [d], Alfonso Senatore [d]

[a] Department of Mathematics and Computer Science, University of Calabria, Rende, Italy
[b] DIMES Department, University of Calabria, Rende, Italy
[c] ASML building 23, HTC 52, High Tech Campus 52, 5656 AG Eindhoven, Netherlands
[d] Department of Environmental and Chemical Engineering, University of Calabria, Rende, Italy

HIGHLIGHTS

- Domain Specific Language for structured grids modeling.
- MPI/OpenMP/OpenCL implementations for execution on heterogeneous systems.
- Efficient built-in data structures and parallel algorithms.
- High computational performances achieved.
- Possibility to devise the best parallel hardware platform for execution.

ARTICLE INFO

Article history:
Received 2 May 2017
Received in revised form 14 February 2018
Accepted 2 July 2018
Available online 17 July 2018

Keywords:
Complex systems modeling
Extended cellular automata formalism
OpenMP
OpenCL
GPGPU computing
MPI

ABSTRACT

This article introduces OpenCAL, a new open source computing abstraction layer for multi- and many-core computing based on the Extended Cellular Automata general formalism. OpenCAL greatly simplifies the implementation of structured grid applications, contextually making parallelism transparent to the user. Different OpenMP- and OpenCL-based implementations have been developed, together with a preliminary MPI-based distributed memory version, which is currently under development. The system software architecture is presented and underlying data structures and algorithms described. Numerical correctness and efficiency have been assessed by considering the *SciddicaT* Computational Fluid Dynamics landslide simulation model as reference example. Eventually, a comprehensive study has been performed to devise the best platform for execution as a function of numerical complexity and computational domain extent. Results obtained have highlighted the OpenCAL's potential for numerical models development and their execution on the most suitable high-performance parallel computational device.

1. Introduction

Scientific Computing [44] is a broad and constantly growing multidisciplinary research field that uses formal paradigms to study complex problems and solve them through simulation by using advanced computing capabilities.

Different formal paradigms have been proposed to provide the abstraction context in which problems are formalized. Partial Differential Equations (PDEs) were probably the first to be largely employed for describing a wide variety of phenomena. Unfortunately, PDEs can be analytically solved only for a small set of simplified problems [56] and Numerical Methods have to be employed to obtain approximate solutions for real situations. Among them, the Finite Differences Method (FDM) was one of the first considered, still currently employed, to address a wide variety of phenomena such as acoustics [17,13], heat [65,69], computational fluid dynamics (CFD) [18,36], and quantum mechanics [47,39].

Besides other solutions proposed for numerically approximating PDEs like, for instance, Finite Elements and Finite Volume Methods, further formal paradigms were more recently proposed for modeling complex systems as result of studies in Computer

* Correspondence to: Department of Mathematics and Computer Science, University of Calabria, Cubo 30B - Via Ponte Pietro Bucci, I-87036 Rende, Italy.
E-mail addresses: donato.dambrosio@unical.it (D. D'Ambrosio), alessio.derango@mat.unical.it (A. De Rango), m.oliverio@dimes.unical.it (M. Oliverio), davide.spataro@asml.com (D. Spataro), william.spataro@unical.it (W. Spataro), rocco.rongo@unical.it (R. Rongo), giuseppe.mendicino@unical.it (G. Mendicino), alfonso.senatore@unical.it (A. Senatore).

https://doi.org/10.1016/j.jpdc.2018.07.005
0743-7315/

Science. Among them, Cellular Automata (CA) [74] are Turing-equivalent [21,22] parallel computational models. CA are widely studied from a theoretical point of view [77,51,78,60], and their application domains vary from Artificial Life [50,10] to Computational Fluid Dynamics [41,57,46,2], besides many others. In the 80s, an extension of the original CA formalism was proposed to better model and simulate a specific set of complex phenomena [45]. Such an extension is known as Complex or Multi-Component Cellular Automata and was applied to the simulation of debris flows [28,6], lava flows [32,33,62,31], pyroclastic flows [5,24], forest fires spreading [4,8], hydrologic and eco-hydrologic modeling [58,66,59,16], soil erosion [27], crowd dynamics [52,76,75], urban dynamics [12], besides others. In this paper we will refer such an extension of the original CA paradigm as Extended Cellular Automata (XCA).

Independently from the adopted formal paradigm, the simulation of complex systems often requires Parallel Computing. OpenMP is the most widely adopted solution for parallel programming on shared memory computers [19]. It fully supports parallel execution on multi-core CPUs and, starting from the 4.0 specification, also includes support for accelerators like graphic processing units (GPUs) or Xeon Phi co-processors. Unfortunately, compilers like gcc currently do not fully support the OpenMP most recent specifications and, in practice, OpenMP-based applications still mainly run on CPUs [62,3,64]. However, in recent years, general purpose computing on graphic processing units (GPGPU), which exploits GPUs and many-core co-processors for general purpose computation, has gained wide acceptance as an alternative solution for high-performance computing, resulting in a rapid spread of applications in many scientific and engineering fields [63]. Most implementations are currently based on Nvidia CUDA (see e.g., [11,29,37,30]), one of the first platforms proposed to exploit GPUs computational power on Nvidia hardware. An open alternative to CUDA is OpenCL [71], an Application Program Interface (API) originally proposed by Apple and currently managed by Khronos Group for parallel programming on heterogeneous devices like CPUs, GPUs, Digital Signal Processors (DSPs), and Field-Programmable Gate Arrays (FPGAs). Interest in OpenCL is continuously growing and many applications can already be found in literature [54,9,38,14]. However, an OpenCL parallelization of a scientific application is often a non-trivial task and, in many cases, requires a thorough re-factorization of the source code. For this reason, many computational layers were proposed, which make many-core co-processors computational power easier to be exploited. For instance, ArrayFire [55] is a mathematical library for matrix-based computation such as linear algebra, reductions, and Fast Fourier transform; clSpMV [72] is a sparse matrix vector multiplication library; clBlas [20] is an OpenCL parallelization of the Blas linear algebra library. Examples of higher level computational layers, which provide the abstraction of formal computational paradigms, are: OPS [68,48] and OP2 [43,67], which are open-source frameworks for the execution of structured and unstructured grid applications, respectively, on clusters of GPUs or multi-core CPUs; AQUAgpusph [15], which is a smoothed-particle hydrodynamics solver; ASL [1], an accelerated multi-physics simulation software based, among others, on the Lattice Boltzmann Method; CAMELot [35,34] and libAuToti [70], which are a proprietary simulation environment and an efficient parallel library for XCA model development, respectively.

In this article we introduce OpenCAL (Open Computing Abstraction Layer), a new open source parallel computing abstraction layer for scientific computing. It provides the Extended Cellular Automata general formalism as a Domain Specific Language, allowing for the straightforward parallel implementation of a wide range of complex systems. Cellular Automata, Finite Differences and, in general, other structured grid-based methods are therefore supported. Different versions of the library allow to exploit both multi- and many-core shared memory devices, as well as distributed memory systems. Specifically, OpenMP- and OpenCL-based implementations have been developed, both of them providing optimized data structures and algorithms to speed-up the execution and allowing for a transparent parallelism to the user. A MPI-based implementation is also currently under development and allows to exploit many-core accelerators on interconnected systems.

Among the above cited software, OPS, ASL and CAMELot probably are the most similar to OpenCAL in terms of modeling and development approach, and could be considered as possible alternatives to the library proposed in this paper. In particular, OPS provides a straightforward Domain Specific Language for structured grid-based modeling, even if it does not refer to any specific abstract computational formalism. Its main characteristic consists in allowing to obtain different parallel versions of a computational model starting from its serial implementation, thanks to a seamless code-generator approach. Both MPI-based distributed memory and CUDA/OpenCL many-core versions can be obtained in this way, with a minimal effort by the developer. Conversely, ASL provides different higher level modeling abstractions among which the Lattice Boltzmann Method, that is eventually a Cellular Automata-based paradigm. Nevertheless, it currently does not allow for parallel execution on distributed memory systems, which can be a great limitation in some cases. Eventually, CAMELot offers an integrated simulation environment for XCA development and allows for parallel execution on both shared and distributed memory systems thanks to the message passing paradigm, not permitting however the exploitation of modern many-core devices. With respect to the above cited software, OpenCAL provides both the higher CAMELot modeling approach and, similarly to OP2, allows for the execution on a wide range of shared and distributed parallel platforms (even if by adopting a classic library approach). In addition, OpenCAL provides different embedded strategies and optimization algorithms which allow to progressively improve the computational performance of different kinds of models and simulations.

In the following, the OpenCAL architecture is presented and the OpenMP- and OpenCL-based parallel implementations described. We also present and discuss the implementation of a first simple example of application for multi- and many-core devices to show how straightforward the OpenCAL-based model development is. We therefore consider the *SciddicaT* XCA landslide simulation model [7] as a more complex reference example for correctness and computational performance evaluation on multi-core CPUs, many-core GPUs, and also on a test multi-node GPU-based system. Specifically, we refer to three different versions of *SciddicaT*, which progressively exploit OpenCAL built-in features and, for each of them, we propose different implementations based on the serial and parallel versions of the library. Eventually, results of a further study performed to devise the best platform for execution, depending on the model's computational intensity and the domain extent, is presented. A general discussion concerning OpenCAL and future outcomes concludes the paper.

2. An OpenCAL overview: Software architecture, main specifications and a first example of application

In this section we describe the software architecture, main structures and underlying algorithms of the OpenCAL library, besides a first example of application to highlight how easy model development is. The serial version of the library will be simply referred as OpenCAL in the following, while OpenCAL-OMP and OpenCAL-CL will refer to the OpenMP- and OpenCL-based parallelizations, respectively. Eventually, the preliminary distributed