Performance Evaluation of the Ocean-Land-Atmosphere Model Using Graphics Processing Units*

Claudio Schepke, Nicolas Maillard
Grupo de Processamento Paralelo e Distribuído (GPPD) – Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
{cschepke,nicolas}@inf.ufrgs.br

Abstract

The Ocean-Land-Atmosphere Model (OLAM) is an atmospheric model to simulate and cover all Earth surface. OLAM demands a great amount of processing in a simulation because of the large number of data structures used to represent the atmosphere. Because of this, we investigate in this paper how to increase performance using GPUs to compute the model. A prototype of OLAM was developed including the call of CUDA threads in order to explore GPU parallelism. Performance evaluations of some atmospheric simulations show that the use of CUDA threads increases significantly the performance of this atmospheric model.

1. Introduction

Numerical models have been extensively used in the last decades to understand and predict weather phenomena and climate, in daily weather forecasts as well as in researches on Global Warming [10], [12]. These models calculate the values of the physical conditions of the atmosphere using quantitative methods. To this end, the atmosphere is represented by a discrete space, a mesh of points obtained through the use of a domain decomposition technique, on which interactions are made during discrete time steps.

A novel interesting approach was recently developed at Duke University. The main feature of this model called Ocean-Land Atmosphere Model (OLAM) [11] is to provide a global grid that can be locally refined, forming a single grid. This feature allows simultaneous representation (and forecasting) of both the global and the local scale phenomena, as well as bi-directional interactions between scales.

In previous works, the parallel performance of the model was provided and evaluated using OpenMP and Message-Passing Interface (MPI) to explore shared and distributed memory systems [9] [8]. The results presented in the works show that it is possible increase performance to the prototype using parallel architectures.

In this work, we use Compute Unified Device Architecture (CUDA) threads in order to explore the parallelism offered by Graphics Processing Units (GPU) architectures [3]. GPU can compute hundreds of concurrently executing instructions simultaneously providing a very high performance. The objective of this paper is evaluate the parallel performance of climatological models computing on GPU. Thus is possible to explore an another kind of parallel architecture.

The remainder of this paper is organized as follows. Section 2 describes the use of GPUs for High Performance Computing. CUDA programming interface is described in Section 3. Section 4 exposes the new prototype implementation of an atmospheric model using CUDA code. The performance evaluation, experimental results and experimental analysis are shown in Section 5. The last section presents the conclusion and the future work.

2. Graphics Processing Units

The performance of contemporary Graphics Processing Units (GPU) has increased much faster than conventional processors, in part because these processors can easily exploit parallelism in the main application areas [6]. Modern GPUs incorporate an array of programmable processors to support the programmable shaders (a set of software instructions) found in graphics APIs [7].

For example, the Nvidia GeForce 9800 includes an array of 128 processors. Each processor can execute only one single-precision floating-point operation in each cycle. This is a significative power processing, because until 128 concurrently instructions can be execute at each clock cycle. Another hardware, the Nvidia Tesla GPU M2090, developed for High Performance Computing, has 512 cores

* Supported by CNPq

and can process 665 Gflops. These two examples show that GPUs can have high levels of parallelism and throughput.

The programmability, high performance, and efficiency of modern GPUs have made them an attractive target for scientific and other non-graphics applications [3]. Programming libraries, such as offered by Nvidia's CUDA, have evolved to support general purpose applications on these platforms [5].

3. Compute Unified Device Architecture

Compute Unified Device Architecture (CUDA) is a parallel programming computing architecture developed by nVidia [5]. It enables the use of GPU, integrated in the video boards, for programming software that use high performance programming architectures. This technology was available initially to the GeForce (series 8 and after) and Quadro editions, and more specifically to the Tesla edition (developed for HPC) and Ion (for mobile computers).

The use of video boards to execute an application normally performed by a CPU is called General-Purpose computing on Graphics Processing Units (GPGPU) [1]. The first advantage of using CUDA is the use of shared memory for quick access arbitrary addresses in memory. Since 3.1 version, CUDA has support for recursion, double data type precision according to the standard IEEE 754, and rendering of textures.

The programming model of CUDA consists of extensions to C and C++ in a sequential program that can boot a kernel [5]. The kernel is a function similar to C and runs in parallel by several threads, which are mapped to the execution core by the own GPU. The programmer is responsible to transfers data from CPU to the GPU.

CUDA programming model is ideal for applications with high data parallelism level and for applications that has not dependencies among tasks. However, CUDA limitations include no control of coherence of the data used and the lack of support for the execution of multiple kernels. Thus, significant performance gains in CUDA depend on good knowledge about the architecture and programming model.

4. Ocean-Land-Atmosphere Implementation

The Ocean-Land-Atmosphere Model (OLAM) is an atmospheric model to simulate and cover all Earth surface. The OLAM model consists essentially of a finite volume representation of the full compressible nonhydrostatic Navier-Stokes equations over the planetary atmosphere with a formulation of conservation laws for mass, momentum, and potential temperature, and numerical operators that include time splitting [4]. The finite volumes are defined horizontally by a global triangular-cell grid mesh and subdivided vertically through the height of the atmosphere forming vertically-stacked prisms of triangular bases.

OLAM was developed and parallelized initially with Message Passing Interface (MPI) [2]. Each OLAM MPI process is responsible for operating the functions of the **iterative step** on a given subdomain. The distribution of data among the processes is set in each one. Each process determines its operating subdomain from the global grid according to its MPI rank. The data distribution takes into account the number of triangular-cell (the mesh points of the domain) resulted of the global domain decomposition, to ensure a good load balance. Once defined the distribution of subdomains among the processes (**initialization step**), each process discards the global mesh, and keeps in memory only its respective part of the global mesh to compute.

After the parallelization with MPI, we provide a CUDA implementation version. OLAM prototype implementation with CUDA involves to rewriting some functions of the C code, converting it to a CUDA kernel code. It was also necessary the implementation of functions that encapsulate allocation, deallocation, and memory copies between CPUs and GPUs. All temporary array variables used in CUDA kernel functions are allocated before the call of the iterative step of the model and released only after all steps of iterative part of the model. This is made to reduce the number of memory allocations in the CUDA kernel.

Moreover, in each iterative step, before each CUDA kernel function call, it is necessary to move data from CPU to GPU and, after the execution of the code, data exchange from GPU to CPU.

CUDA kernel functions was also embedded in a MPI implementation. In this way, GPUs and multi-processors parallelism can be employed for an atmospheric simulation.

5. Performance Evaluation

This section presents the simulation environment, execution parameters, and execution time measurements.

5.1. Simulation Environment

All experimental measurements were obtained using the *Newton* cluster of the *Centro Nacional de Supercomputação*. This cluster is interconnected by an InfiniBand network technology, and has currently 28 Sun Fire X2200+ workstations (each on with 2 Quad-Core AMD Opteron 2.2 GHz processors and 16 GB RAM) and a coupled performance of 1.97 TFlops; and 8 GPUs nVidia Tesla S1070 with coupled performance of 8.28 TFlops.

For parallel executions, the processes are balancing distributed among the GPUs. The number of CUDA threads was fixed in 128 for each GPU.

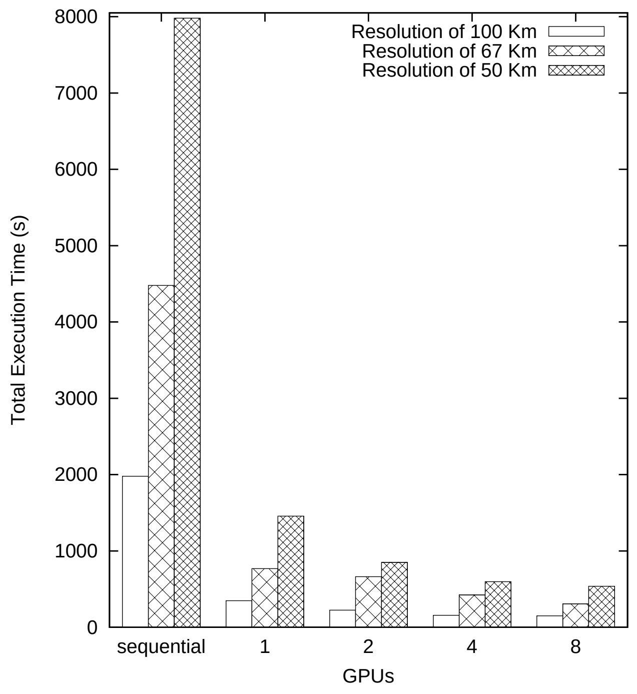

Figure 1. Execution time evaluation using different number of GPUs

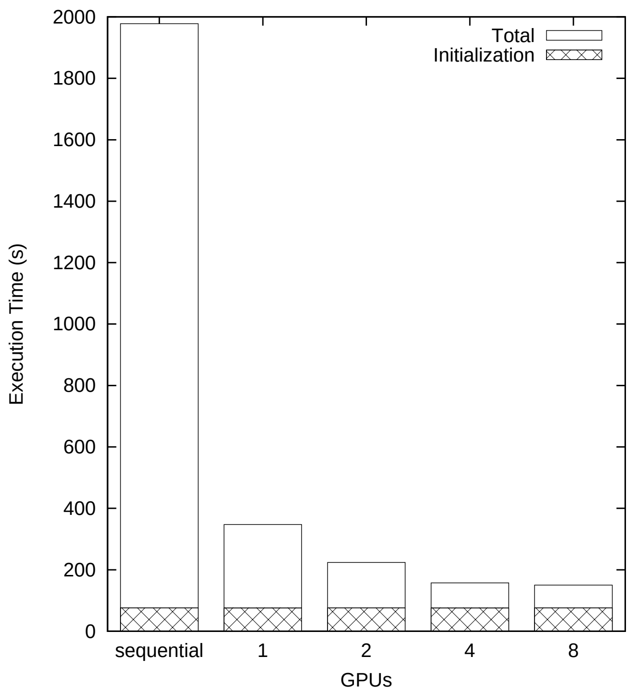

Figure 2. Initialization and iterative step execution time for a 100 Km of mesh resolution

In all executions, we simulate 12 hours of integration of the atmosphere. Each timestep of integration simulates 60 seconds of the real time of the weather condition. The vertical axis of the atmosphere was divided in 28 layers. The distance between each pair of points on the globe surface was around 100 Km, 67 Km and 50 Km. Each MPI process runs over a GPU.

5.2. Execution Time

Figure 1 presents the total execution time (in seconds) for the three resolutions considered in this work (see Subsection 5.1). In this figure, the first three columns show the sequential execution time. The sequential execution time not include the use of GPUs. The other columns of the graphic present the execution time using 1, 2, 3, and 8 GPUs.

The results of Figure 1 show that the use of one GPU reduces the total execution time more than 5 times in relation to the execution using only CPU processing. This reduction is more expressive as more GPUs are used in the simulations.

Figure 2, and Figure 3 presents the execution time (in seconds) of the initialization and iterative step of the model for a mesh resolution of 100 Km, and 50 Km, respectively. In these results is possible to see that the execution time for the initialization step is constant independently of the number of GPUs used. On the other hand, the execution time of the iterative step decreases as more GPUs are included in the computation in all cases evaluated.

Figure 4 shows the speed up of the iterative step of the model using 1 to 8 GPUs in simulations with mesh resolution of 100 Km, and 50 Km. In all mesh resolution cases the speed up increases as more GPUs are used. The graphic shows also that a mesh with high resolution (50 Km) has more speed up as a mesh with low resolution (100 Km) because the difference granularity among the processes. High mesh resolutions have more data structures to compute. Thus, the granularity of the processes is larger in this case.

5.3. Conclusion and Future Works

This paper presented a parallel implementation of the Ocean-Land-Atmosphere Model using CUDA programming interface. We evaluate the performance of a prototype version of this atmospheric model in GPUs architecture.

In order to evaluate the implementation, we present partial and comparative execution time and speed up using 1 to 8 GPUs. The partial measurement results show that the CUDA implementation version increases 5 times more performance in relation to a CPU implementation version.

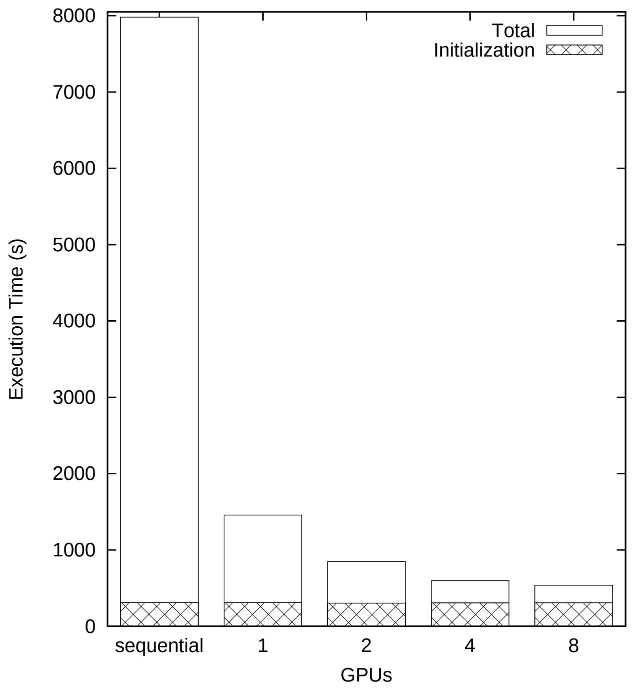

Figure 3. Initialization and iterative step execution time for a 50 Km of mesh resolution

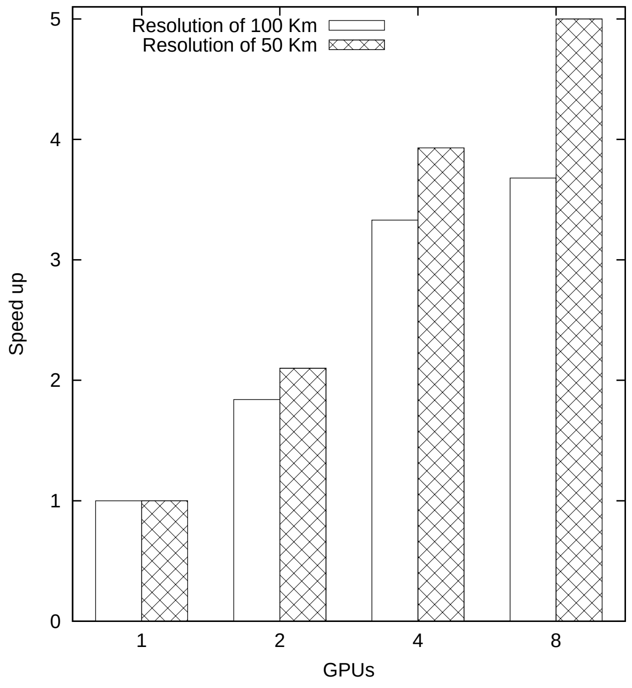

Figure 4. Speed up evaluation using different number of GPUs

We also evaluate the performance of the prototype computing in more than one GPU. The results shown that there are an increase of speed up as more GPUs are used in all mesh configurations adopted in the simulations.

We are planning to evaluate the OLAM prototype in GPU architectures changing the number of CUDA threads. This number is currently fixed in 128 threads. More threads can increase the performance in some mesh configurations.

References

[1] M. Garland, S. L. Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov. Parallel Computing Experiences with CUDA. *IEEE Micro*, 28(4):13–27, 2008.

[2] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. High-performance, portable implementation of the MPI Message Passing Interface Standard. *Parallel Computing*, 22(6):789–828, 1996.

[3] D. B. Kirk and W. m. W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, San Francisco, CA, USA, 1st edition, February 2010.

[4] J. Marshall, A. Adcroft, C. Hill, L. Perelman, and C. Heisey. A Finite-Volume Incompressible Navier-Stokes Model for Studies of Ocean on Parallel Computers. *Journal of Geophysical Research*, 102(C3):5753–5766, 1997.

[5] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable Parallel Programming with CUDA. *Queue*, 6(2):40–53, 2008.

[6] J. Nickolls and W. Dally. The GPU Computing Era. *IEEE Micro*, 30(2):56–69, mar. 2010.

[7] Nvidia. High Performance Computing - Supercomputing with Tesla GPUs, Jun. 2012. Available at: <http://www.nvidia.com/object/tesla_computing_solutions.html>. Last access: June, 2012.

[8] C. Schepke, N. Maillard, C. Osthoff, and P. Dias. Performance Evaluation of an Atmospheric Simulation Model on Multi-Core Environments. In *Proceedings of Conferencia Latino Americana de Computación de Alto Rendimiento*, pages 330–332, Gramado, RS, Brazil, 2010. Instituto de Informtica/UFRGS.

[9] C. Schepke, N. Maillard, J. Schneider, and H.-U. Heiss. Why Online Dynamic Mesh Refinement is Better for Parallel Climatological Models. In *23th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2011)*, Vitória, Espírito Santo, 2011. IEEE.

[10] T. Vasquez. *Weather Forecasting Red Book*. Weather Graphics Technologies, Garland TX, USA, 2006.

[11] R. L. Walko and R. Avissar. The Ocean-Land-Atmosphere Model (OLAM). Part I: Shallow-Water Tests. *Monthly Weather Review*, 136(11):4033–4044, 2008.

[12] W. M. Washington and C. L. Parkinson. *An Introduction to Three Dimensional Climate Modeling*. University Science Books, Herndon, VA, USA, 2 edition, 2005.