Programming Models case studies on Atmospheric Model Application

Roberto P. Souto Carla Osthoff Fabrício Vilasbôas Arthur B. Oliveira Pedro L.S.Dias

Laboratório Nacional de Computação Científica – (LNCC), Brazil

Abstract

This work discuss the parallel performance of a global numerical simulation model, Ocean-Land-Atmosphere Model (OLAM), on a hybrid multicore/GPU environment, under the following programming models: (1) a parallel MPI implementation running one process per core on the multi-core node, (2) a parallel OpenMP implementation running one thread per core on the multi-core node, (3) a hybrid parallel OpenMP/OpenAcc implementation running OpenMP threads on the cores and OpenAcc/Cuda threads on the GPU and (4) a hybrid parallel OpenMP/Cuda implementation running OpenMP threads on the cores and Cuda threads on the GPU. The results show that the adopted programming model impacts significantly the performance of the application. We show that as we increase the number of cores, the parallel MPI implementation running processes on each core executes faster them other implementations.

Keywords: Multi-core, GPU, Atmospheric Numerical Simulation Model, High Performance Computing

Authors' contact:
{rpsouto,osthoff,fabricio, arthurbo, pldsdias}
@lncc.br

1. Introduction

Numerical models have been used extensively in the last years in order to understand and predict the climate and weather phenomena. In general, two approaches were followed for the development of numerical models: global and regional. Global models have spatial resolution of about 2-5 degrees of latitude and therefore can´t represent very well the scale of regional weather phenomena. To represent all meteorological processes is necessary a large number of points on the grid. Consequently, this requires a large power processing capacity. Moreover, regional models have more accurate resolution, but are used for specific regions. It must integrate large-scale atmospheric conditions into its borders side. Therefore it does not simulate the phenomena of large scale.

To solve this problem recently was developed at Duke University a new model that represents a new generation of meteorological models. The main feature of this model called Ocean-Land Atmosphere Model (OLAM) is the ability to represent global phenomena weather and also allows grids nesting with high resolution enabling more accurate representation of the phenomena of local scale [Walko and Avissar].

Due to the large computational demands and execution time constraints, these models rely on parallel processing. They are executed on clusters or grids in order to benefit from the architecture's parallelism and divide the simulation load. On the other hand, over the next decade the degree of on-chip parallelism will significantly increase and processors will contain tens and even hundreds of cores, increasing the impact of levels of parallelism on clusters. Legacy applications and research efforts that do not invest in multi-threaded software will likely not benefit from modern multi-core processors because single-threaded and poorly scaling software will not be able to utilize extra processor cores.

Today general purpose computation on Graphics Processing Units, GPGPU (2007), is a trend that uses GPU for general-purpose computing. The modern GPU' highly parallel structure makes them often more effective than general. This work discusses the parallel performance for Atmospheric Simulation Model Application in a hybrid system with multicore processors and GPU devices for distinct programming models.

2. Related Work

In [Michalakes et al., 2008] the authors execute the high resolution WRF weather forecast code over 15k cores and conclude that one of the greatest bottlenecks is data storage. The work of [Wolfe, 2009] presents the lessons learned from porting a part of the Weather Research and Forecasting Model (WRF) to the PGI Accelerator. Like OLAM, the application used in our work, the WRF model is a large application written in FORTRAN. They ported the WSM5 ice microphysics model and measured the performance. This measurement compared the use of PGI Accelerator with a multi-core implementation and with a hand-written CUDA implementation. Finally, the work of [Govett10] runs the weather model from the Earth System Research Laboratory (ESLR) on GPUs and relies on the CPUs for model initialization, I/O and inter-processor communication. They have shown that the part of the code that computed dynamics of the model runs 34 times faster on a single GPU than on the CPU.

2. OLAM Typical simulation Analysis

Our case study replicates a typical global forecast with 40 km horizontal resolution, requiring the subdivision of each Olam grid icosahedron edge in 25 x 25 parts. The atmospheric layer (z dimension) was divided in 28 layers. We simulate 24 hours of integration of the equations of atmospheric dynamics without any additional physical calculation (such as moisture and radioactive processes) because we have interest only in the impact on the cost of fluid dynamics executions and communications. Each integration timestep simulates 60 seconds of the real time. This typical simulation requires reading 10GB of input files and initial conditions. OLAM input files are not partitioned for parallel processing. Typical input files are: global initial conditions at a certain date and time and global maps describing topography, soil type, ice covered areas, Olson Global Ecosystem (OGE) vegetation dataset, depth of the soil interacting with the root zone, sea surface temperature and Normalized Difference Vegetation Index (NDVI).

After reading the input file, the processing and data output phases are executed alternately: during each processing phase, OLAM simulates a number of timesteps, evolving the atmospheric conditions on time-discrete units. After each timestep, processes exchange messages with their neighbors to keep the atmospheric state consistent. This is done in an asynchronous manner to hide the cost of message transmission. After executing a number of timesteps, the variables representing the atmosphere are written to a history file. During this phase, each process opens the history file for that superstep, writes the atmospheric state and closes the history file. These history files are considered of small size for the standards of scientific applications. We divide OLAM algorithm in three major parts: the initialization, the atmospheric time state calculation and the output.

3 Performance Evaluation

Figure 1 presents the performance evaluation of the four versions : 1) OLAM MPI implementation, on the multicore system, starts one MPI process on each core of the platform, 2) OLAM OpenMP implementation developed by RobertWalko. This implementation starts OpenMP threads on the cores of the node, This implementation uses Portland C and Fortran compiler OpenMP version 3.0 directives. 3) a hybrid parallel OpenMP/OpenAcc implementation running OpenMP threads on the cores and threads on the GPU developed by Roberto P. Souto. This implementation uses Portland C and Fortran OpenACC directives. (4) a hybrid parallel OpenMP/Cuda implementation running one process per core and Cuda threads on the GPU device developed by Roberto P. Souto.

The performance measurements were made on a multi-core/manycore system, denoted prjCUDA, located at the Brazils National Laboratory of Scientific Computing (LNCC), composed of a dual Hexacore Xeon E5650, 2.67GHz, with 12 MB of L3 cache and 24 GB of RAM memory, GTX285 and Tesla C2050. The software employed included MPICH version 2-1.2.p1, Vtune Performance Analyzer version 9.1, CUDA toolkit version 4.0 and PGI FORTRAN version 11.2 compiler. The experiments evaluate parallel performance of the four implementations of OLAM. As we are running in one single node system, OLAM input and output phase execution times are the same for all implementations, therefore we are not using them for our analysis.

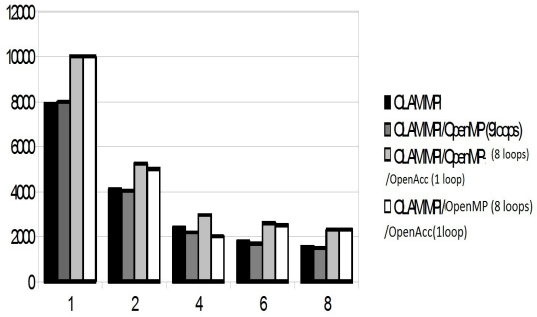

Figure 1 OLAM programming models execution time as increases the number of cores.

References

GOVETT, M. ET AL., RUNNING THE NIM NEXT-GENERATION WEATHER MODEL ON GPUS. PROCEEDINGS OF 10TH IEEE/ACM INTERNATIONAL CONFERENCE ON CLUSTER, CLOUD AND GRID COMPUTING. MELBORNE, AUSTRALIA, PAGES 729-796,2010.

MICHALAKES, J., HACKER, J., LOFT, R., MCCRACKEN, M. O., SNAVELY, A., WRIGHT, N. J., SPELCE, T., GORDA, B., & WALKUP, R. WRF NATURE RUN. JOURNAL OF PHYSICS: CONFERENCE SERIES 125(1):012022, URL HTTP://STACKS.IOP.ORG/ 1742-6596/125/I=1/A=012022.

OSTHOFF, C., VILASBOAS, F., SOUTO, R.P., J., GRUNMANN, P., SILVA DIAS, P. L., BOITO,F., KASSICK, R., PILLA, L., SCHEPKE, C., MAILLARD, N., NAVAUX,P., PANETTA, J., LOPES, P.P. AND WALKO, R. I/O PERFORMANCE EVALUATION ON MULTICORE CLUSTERS WITH ATMOSPHERIC MODEL ENVIRONMENT. ATMOSPHERIC MODEL APPLICATIONS, INTECH OPEN SCIENCE BOOK, ISBN 978-953-51-0488, 2012.

WALKO, R.L. AND AVISSAR, R. OLAM: OCEAN-LAND-ATMOSPHERE MODEL - MODEL INPUT PARAMETERS - VERSION 3.0. TECH. REP., DUKE UNIVERSITY , NOVEMBER, 2008.

WOLFE, M. THE PGI ACCELERATOR PROGRAMMING MODEL ON NVIDIA GPUS PART 3: PORTING WRF. IN THECHNICAL NEWS FROM PORTLAND GROUP. HTTP://WWW.PGROUP.COM/LIT/ARTICLES/INSIDER/V1N3A1.HTM