OLAM Performance Evaluation on Multi-Core Environments*

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
WWW home page: http://gppd.inf.ufrgs.br/atmosferamassiva

Abstract

Ocean-Land-Atmosphere Model (OLAM) is a global numerical simulation model developed at Duke University, USA. It introduces a new dynamic core in the atmospheric model based on a global geodesic grid with triangular mesh cells and uses also a finite volume discretization of the full compressible Navier Stokes equations. Recently this model was parallelized using Single Program Multiple Data (SPMD) approach. OLAM is a typical example of application of domain decomposition that frequently occurs in many areas of the science. Moreover, this real application has a high load computation, being a good candidate for evaluate multi-core cluster environments. In this context, this paper discusses the parallel performance of OLAM implementation in multi-core environments. The results show that the execution in multi-core systems impact significantly in the performance of the application, obtaining less performance than single core systems.

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 [3]. 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. Moreover, this models must also integrate large-scale atmospheric conditions into its borders side. Therefore it does not simulate 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 phenomena of local scale [5].

OLAM was developed to extend features of the Regional Atmospheric Modeling System (RAMS) to a global model domain [3]. OLAM uses many functions of RAMS, including physical parametrization, data assimilation, initialization methods, logic and coding structure, and I/O formats [4]. OLAM introduces a new dynamic core based on a global geodesic grid with triangular mesh cells [6]. It use also a finite volume discretization of the full compressible Navier Stokes equations [2]. OLAM was developed in FORTRAN 90 and recently parallelized with Message Passing Interface (MPI) [1]. to Single Program Multiple Data (SPMD) model.

The remainder of this article is divided in four sections. Next section presents the OLAM global domain grid formation, algorithm and parallelization. A parallel performance evaluation executed in a multi-core/multi-computer system is presented in Sec. 3. Some considerations are described at the last section, as well the conclusions and future works.

2. Ocean-Land-Atmosphere Model

2.1. Global Grid Structure

.

OLAM's global computational mesh consists of spherical triangles, a type of geodesic grid that is a network of arcs that follow great circles on the sphere [5]. OLAM's grid construction begins from an icosahedron inscribed in the spherical earth. Icosahedron is a regular polyhedron that consists of 20 equilateral triangle faces, 30 triangle edges, and 12 vertices, with 5 edges meeting at each vertex. The icosahedron is oriented such that one vertex is located at each geographic pole, which places the remaining 10 vertices at latitudes of $\pm tan^{-1}(1/2)$.

* Supported by CNPq

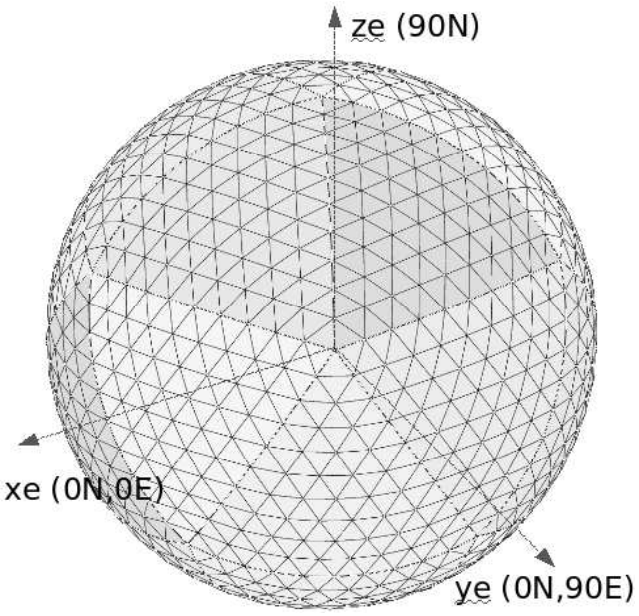

Figure 1. OLAM subdivided icosahedral mesh and cartesian coordinate system with origin at Earth center.

Uniform subdivision of each icosahedron triangle into $N \times N$ smaller triangles, where N is the number of divisions, is performed in order to construct a mesh of higher resolution to any degree desired. The subdivision adds $30(N^2 - 1)$ new edges to the original 30 and $10(N^2 - 1)$ new vertices to the original 12, with 6 edges meeting at each new vertex. All newly constructed vertices and all edges are then radial projected outward to the sphere to form geodesics.

Figure 1 shows an example of the mesh at this step with $N = 10$. Heavier lines denote original undivided icosahedron. The projection causes most triangles to deviate from equilateral shape, which is impossible to avoid [5].

The final step of the mesh construction is the definition of its vertical levels. To do this, the lattice of surface triangular cells is projected radially outward from the center of the earth to a series of concentric spheres of increasing radius This creates prism-shaped grid cells having two horizontal faces (perpendicular to gravity) and three vertical faces. The horizontal cross section of each grid cell and column expands gradually with height.

OLAM uses a rotating Cartesian system with origin at the Earth's center, z-axis aligned with the north geographic pole, and x- and y-axes intersecting the equator at 0 deg and 90 deg E. longitude, respectively. The three-dimensional geometry of the mesh, relating to terms in the momentum equation and involving relative angles between proximate grid cell surfaces, is worked out in this Cartesian system.

3. Algorithm

OLAM algorithm can be divided in three major parts; the parameter initialization , the atmosphere time state calculation and the output write results. In order to find the algorithm routine that more impact in the execution time, we inserted timestamps barriers on the routines of the code. We selected major 7 timestamps ($TS1$ to $TS7$) that dominates the algorithm overhead. Next we present OLAM algorithm pseudo code and the selected timestamps barriers.

```
Initialization;
Input Files (ATM/LAND/SEA) Read; (TS1)
Grid Configuration;
Domain Partition Decomposition; (TS2)
Variables Memory Allocation; (TS3)
Initial state calculation; (TS4)
Plot/History Files Initialization; (TS5)
Initialization Time measure;
Do loop for each time step;
  Atmosphere time state calculation;
  Send frontier variables to neighbors;
  Times step Time measure;
  If time equal END then;  (TS6)
    End Do Loop;
Write atmosphere state on disk; (TS7)
Barrier; Output Time measure;
```

3.1. Parallelization of the Model

OLAM was developed in FORTRAN 90 and recently parallelized with Message Passing Interface (MPI) [1] to Single Program Multiple Data (SPMD) model.

All MPI processes have initially the original grid domain and it data structures created as described in Section 2.1. In a second moment, if the execution is set as parallel, each process defines recursively his sub-domain. Data are reallocated after the definition of the sub-domain in each process, so that only the sub-domain is keep.

After the parallel grid domain decomposition and data structures redefinition will be process the iterative step. In the iterative step there are data exchange among the processes through asynchronous messages to update physical properties in neighbor processes.

4. Performance Evaluation

4.1. Simulation Environment

All measurements have been made on the cluster ICE, at the Institute of Informatics of the Federal University of Rio Grande do Sul. This cluster is composed by 14 dual nodes Intel Xeon E5310 Quad-Core of 1.6 GHz and 4 MB of cache, with 16 GB of RAM memory in each node.

We divide each side of the initial icosahedral triangle in 25 parts. So, the distance among the points on the globe surface was around 200 Km. The atmosphere layer was divided in 28 layers. We simulate 24 hours of integration of an atmosphere without any physical calculation. Each timestep of integration simulate 60 seconds of the real time.

4.2. Scalability

The first test realized consist in evaluate the scalability of the code in a multi-core machine. In Figure 2 is presented the speed up obtained using all 8 cores of two processors in a node of the cluster and the speed up resulted of using only a core of all 14 nodes of the cluster. The results show a increase of performance when more processes are used in both cases. However, the scalability is less expressive using only a node of the cluster. In fact, the speed up using all 8 cores of one node is only around 5. On the other hand, executing in only 1 core of all nodes of the cluster resulted more scalability. In this case the speed up with 14 processes was up to 11.

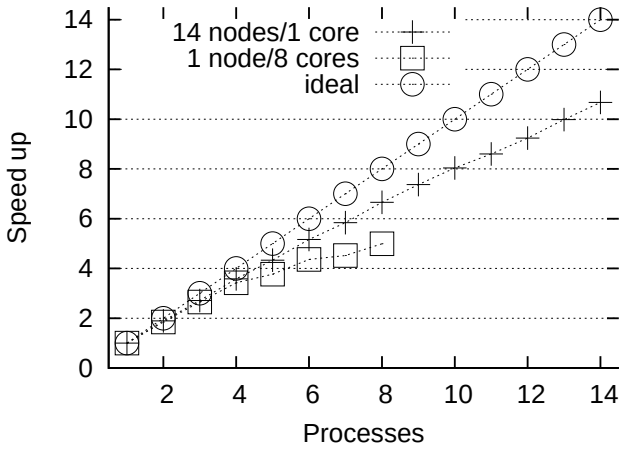

Figure 2. Speed up using 1 node/8 cores $\times$ 8 nodes/1 core.

A more specific evaluation was made to better understand this results. In Table 1 are related the best (T. Min) and worst (T. Max) execution time in milliseconds (ms) of 7 parts of the instrumented code when 8 processes are executed in one node and when 8 processes are executed in distinct nodes, respectively.

Each timestamp (TS) stage $TS1, ..., TS7$ is following by a synchronization stage. The synchronization stage measures the time elapsed between the end of each execution time stage and a barrier, where all processes must arrive.

Time-Stamp	T. Min 1 node	T. Max 1 node	T. Min 8 nodes	T. Max 8 nodes
TS1	598	618	180	182
Syn	0	31	0	6
TS2	186	225	58	59
Syn	0	39	0	1
TS3	20	23	15	17
Syn	0	39	0	3
TS4	565	601	554	555
Syn	0	1	0	0
TS5	109	280	103	266
Syn	0	170	0	163
TS6	111268	111924	81641	82194
Syn	83	837	129	704
TS7	173	487	132	138
Syn	109	585	130	302
Total	**114497**	**114510**	**83830**	**83837**

Table 1. Execution time using 8 processes in 1 node and in 8 nodes.

In this table we can observe that timestamp 6 is the most impact step of the execution. This timestamp monitors the iterative step of OLAM and in this case represent around 97% of the total execution time in both cases evaluate. However, the difference between the cases 1 node $\times$ 8 nodes is very significant.

4.3. Execution Time

We distribute P processes to C cores processors to better understand how the cores of the architecture influence the execution of OLAM. Thus P/C nodes will be used to execute the model.

Figure 3 shows a comparison of the parallel execution time of OLAM, distributing 14 processes among the ICE cluster nodes in five different ways. These ways consist in to distribute the processes respectively in one, two, four, six and eight cores per processor ($C = 1, 2, 4, 6, 8$).

The results show that using only a core per node is better than use more cores per node. The results also demonstrate that performance decreases as the number of cores used increases. This is more visible when more than 4 cores per node are used. In fact, quad-core processors share the bus access. Because this, the performance on access simultaneously the memory is not so good, instead large volume of data are manipulated in OLAM code.

In Table 2 are related the best and worst execution time in milliseconds (ms) of each of the instrumented code when

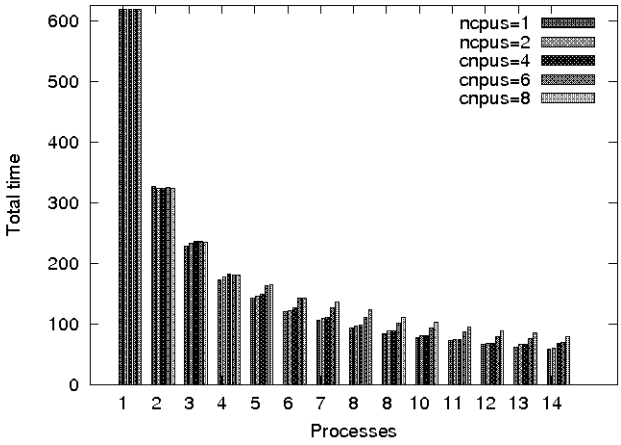

Figure 3. Execution time distributing the processes in 1, 2, 4, 6 and 8 cores.

14 processes are executed using 1 core $C = 1$ and 8 cores $C = 8$ of the cluster nodes.

Observe that the synchronization time from $TS6$ and $TS7$ dominates the overhead for both $C = 1$ and $C = 8$. Also, observe that the synchronization time increases for all timestamps for the $C = 8$ test showing that the use of all cores in a node impact in reduction of performance.

Synchronization time from $TS6$ is related to the load imbalance from the atmosphere time state calculation part.

Time-Stamp	T. Min $C = 1$	T. Max $C = 1$	T. Min $C = 8$	T. Max $C = 8$
TS1	178	183	177	1095
Syn	0	149	0	959
TS2	57	60	55	201
Syn	1	4	0	147
TS3	9	13	15	18
Syn	1	5	0	38
TS4	323	367	339	374
Syn	0	1	0	8
TS5	88	195	93	191
Syn	0	107	0	90
TS6	47323	49249	70063	72481
Syn	244	2224	463	2862
TS7	43	134	121	175
Syn	254	510	513	964
Total	**49884**	**50786**	**75496**	**75552**

Table 2. Execution time using 14 processes which $C = 1$ and $C = 8$.

Synchronization increase time from $TS1$ to $TS7$ on $C = 8$ test are related to the multi-core memory contention. Applications on systems with a large numbers of multi-core processors, with a meager amount of local RAM per core generally use the majority of this memory, leaving little room for caching

5. Conclusion

This paper evaluates the performance of the parallelized version of the Ocean-Land-Atmosphere Model (OLAM) on a multi-core cluster environment. In order to evaluate the scalability we present the speed up obtained using all 8 cores of two processors in a node of the cluster and the speed up resulted of using only a core of all 14 nodes of the cluster. The results indicate that the memory access is limited in a multi-core system. We conclude that using many cores of a multi-core system results in competition to access the memory and more cache misses.

In order to find OLAM algorithm routines that increase the execution time with the number of processors, we inserted timestamps barriers on routines of the code. We observe that the routines that dominate OLAM application execution time are related to OLAM model timestep calculation and to output write operations. The results indicate that the barrier synchronization time increases as we increase the number of cores per node.

In future works more tests will occur, including more mesh refinement and inclusion of a domain decomposition distribution controller.

References

[1] 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.

[2] 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.

[3] R. A. Pielke and et al. A comprehensive meteorological modeling system-RAMS. *Meteor. Atmos. Phys.*, 49:69–91, 1992.

[4] R. L. Walko and R. Avissar. OLAM: Ocean-Land-Atmosphere Model - Model Input Parameters - Version 3.0. Technical report, Duke University, November 2008.

[5] 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.

[6] I. Wenneker, A. Segal, and P. Wesseling. A mach-uniform unstructured staggered grid method. *International Journal of Numerical Methods in Fluids*, 40(9):1209–1235, 2002.