Int J Parallel Prog (2013) 41:552–569
DOI 10.1007/s10766-012-0235-4

Online Mesh Refinement for Parallel Atmospheric Models

Claudio Schepke · Nicolas Maillard · Joerg Schneider · Hans-Ulrich Heiss

Received: 29 November 2011 / Accepted: 22 November 2012 / Published online: 5 December 2012

Abstract Forecast precisions of climatological models are limited by computing power and time available for the executions. As more and faster processors are used in the computation, the resolution of the mesh adopted to represent the Earth's atmosphere can be increased, and consequently the numerical forecast is more accurate. However, a finer mesh resolution, able to include local phenomena in a global atmosphere integration, is still not possible due to the large number of data elements to compute in this case. To overcome this situation, different mesh refinement levels can be used at the same time for different areas of the domain. Thus, our paper evaluates how mesh refinement at run time (online) can improve performance for climatological models.The online mesh refinement (OMR) increases dynamically mesh resolution in parts of a domain,when special atmosphere conditions are registered during the execution. Experimental results show that the execution of a model improved by OMR provides better resolution for the meshes, without any significant increase of execution time. The parallel performance of the simulations is also increased through the creation of threads in order to explore different levels of parallelism.

C. Schepke (✉)· N. Maillard
Programa de Pós-Graduação em Computação (PPGC), Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS), Caixa Postal 15.064, 91.501-970 Porto Alegre, RS, Brazil
e-mail: claudioschepke@unipampa.edu.br

N. Maillard
e-mail: nicolas@inf.ufrgs.br

J. Schneider · H. U. Heiss
Fachgebiet Kommunikations- und Betriebssysteme (KBS), Institut für Telekommunikationssysteme, Technische Universität Berlin (TU-Berlin), Einsteinufer 17, 10587 Berlin, Germany
e-mail: komm@cs.tu-berlin.de

H. U. Heiss
e-mail: heiss@cs.tu-berlin.de

Keywords Atmospheric models · Online mesh refinement · Parallel applications · High performance computing

1 Introduction

Numerical models have been used extensively in the last decades to understand and predict weather phenomena and the climate, in daily weather forecasts as well as in researches on Global Warming [17].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.

As the domain refinement increases, more points are used in the mesh representation, and consequently forecasts become more accurate. Therefore, the impact of various physical factors, that vary in a continuous space, are more visible and taken into account during the simulation [19].

Atmospheric models generally define the mesh at the beginning of the execution, before the calculation of the physical properties at the iterative step, in a static approach [13]. For long numerical simulations it is important that the mesh resolution can be adapted while the code is running. Thus, spontaneous atmospheric changes that appear in restricted areas, for a given time during the execution, like storms and hurricanes, can be better investigated applying more mesh resolution in part of the domain. At the same time, their impact in the whole mesh domain can be understood better.

A previous work [15], has detailed a mechanism to refine the mesh in climatological models at execution time. It allows a part of the mesh to increase its resolution when special atmosphere conditions appear during the simulation. However, the refinement suffered from load unbalance. In this new work, we build upon [15] to improve the evaluation of the impact of the online mesh refinement (OMR) on the parallel performance, and propose solutions using threads to balance the load and explore different levels of parallelism. This new solution leads to an increased speed-up.

The remainder of this paper is organized as follows. Section 2 presents the related work. In Sect. 3 we compare the difference between global and local climatological models. The static mesh refinement of an atmospheric model is described in Sect. 4. Section 5 discusses the necessity and the problems associated to the use of high mesh resolutions in atmospheric models. Section 6 exposes the distributed OMR implementation. The performance evaluation, experimental results and experimental analysis are shown in Sect. 7. Section 8 describes a solution to the load unbalance problem of the OMR implementation and it impact in the performance in the atmospheric model. The last section presents the conclusion and the future work.

2 Related Work

Mesh refinement at runtime is not a new idea to improve performance for a decomposed domain. The adaptive mesh refinement (AMR) technique is frequently cited in the literature as a way to represent complex geometry and to increase locally the resolution for a thin part of a domain [12]. This technique is used in computational fluid dynamics

to add fine grid patches to regions of the flow where more resolution is needed, such as near shocks and detonations.

The AMR can dramatically speed up a computation and/or enable simulations with a much higher effective resolution as compared to the uniformly refining of the grid approach. Efficient numerical schemes can be written for overlapping grids since they are composed of structured grids and Cartesian grids.

There are many applications developed using this technique. PARAMESH is an example of toolkit designed to provide parallel support with adaptive mesh capability for a large and important class of computational models, those using structured logically Cartesian meshes [8]. The PARAMESH package of subroutines is designed to provide an application developer an easy way to extend an existing serial code into a parallel code with adaptive mesh refinement.

However, mesh refinement solutions, like PARAMESH, are restricted to structured grids and differ from the unstructured mesh adopted in many climatological models.

3 Atmospheric Models

There are several climatological models developed today, each one implemented for a specific proposal. In general, according to the form to represent the atmosphere, these models can be classified in two categories, differing on their domain: global (entire Earth) and regional (country, state, etc).

Global models, like GISS ModelE [16], consider the entire surface of the Earth for modeling and decomposing the domains and are usually used to predict long climatological periods (months, years). The main limitation of this approach is the computing power to execute with higher mesh resolution. Global models have normal spatial mesh resolution of about 0.2 to 1.5 degrees of latitude and therefore cannot represent very well the scale of regional weather phenomena.

On the other hand, **Regional models**, like BRAMS [4], simulate only a specific interesting piece of the Earth atmosphere. They use higher mesh resolution but they are restricted to limited areas. Therefore, it is necessary to establish the initial entry conditions to the boundary of the domain. These conditions can be determined from previous executions of global models.

The way to use the best characteristics of both approaches is offering different levels of mesh refinement in global models. Thus, it is not necessary to handle boundary conditions, since the transition among different levels of refinement is done by a transparent design.

This is the case of the ocean-land atmosphere model (OLAM) [18], which provides a global grid that can be locally refined, forming a single grid. This feature allows simultaneous representation (and forecasting) of both, global and local scale phenomena, as well as bi-directional interactions between scales.

4 Ocean-Land-Atmosphere Model

The 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 [9]. 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.

4.1 Global Grid Structure

OLAM's grid construction begins from an icosahedron inscribed in the spherical Earth, as is the case for most other atmospheric models that use geodesic grids. The geodesic grid offers important advantages over the commonly used latitude-longitude grid. It allows mesh size to be approximately uniform over the globe, and avoids singularities and grid cells of very high aspect ratio near the poles.The icosahedron is oriented such that one vertex is located at each geographic pole, which places the remaining 10 vertices at latitudes of $\pm tan^{-1}(1/2)$.

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 radially projected outward to the sphere to form geodesics.

Figure 1 shows an example of the mesh at this step. The projection causes most triangles to deviate from equilateral shape, which is impossible to avoid [18].

OLAM uses an unstructured approach and represents each grid cell with single horizontal index [18]. Required information on local grid cell topology is stored and accessed by means of linked lists.

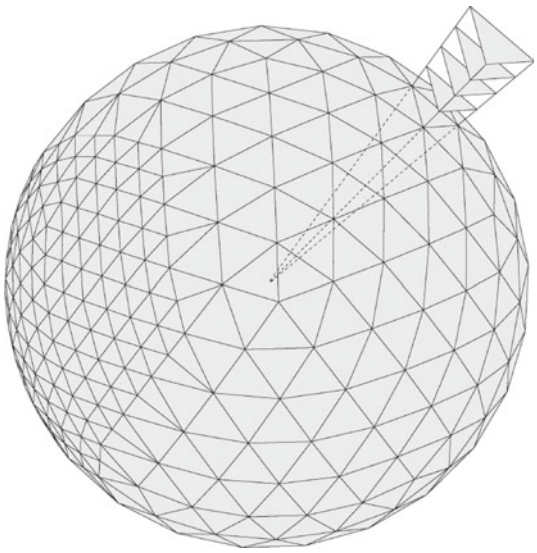

Fig. 1 Projection of a surface *triangle* cell to larger concentric spheres to generate multiple *vertical* model levels

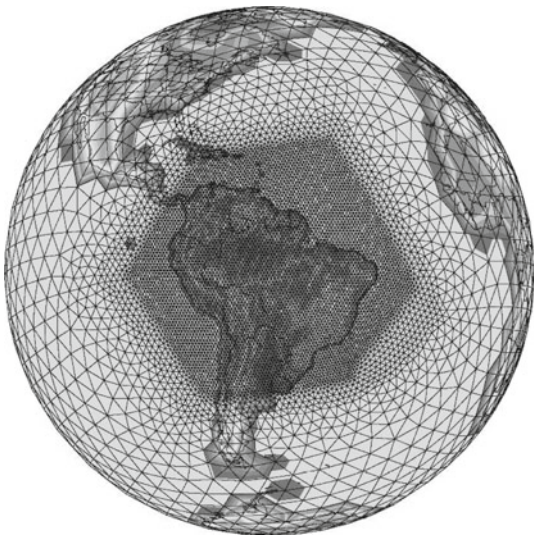

Fig. 2 Example of local mesh refinement applied to a selected part of the globe

4.2 Static Local Mesh Refinement

Local refinement can be specified to cover specific geographic areas with higher resolution. The mesh points that represent these areas are subdivided cyclically while the expected mesh resolution is not achieved. Each cyclical division doubles the resolution.

The global grid and its refinements define a single grid, as opposed to the usual nested grids of regional models. Grid refined cells do not overlap with the global grid cells - they substitute them.

Refinement follows a three-neighbors rule that each triangle must share finite edges length with exactly three others.

An example of local mesh refinement is illustrated in Fig. 2, where the resolution is exactly twice that of the original resolution. This is achieved by subdividing each previous triangle into 2 smaller triangles. For this purpose, auxiliary edges were inserted at the boundary between the original and refined regions in order to preserve the rule of the three neighboring triangles for each triangle.

A transition from coarse to fine resolution is achieved by using vertices with more than 6 edges on the coarser side and vertices with fewer than 6 edges on the finer side of the transition as can be seen in Fig. 3. In this example each auxiliary line connects a vertex that joins 7 edges with a vertex that joins 5 edges. However, it is not necessary that these vertices are concentrated along a band. A more gradual refinement of the mesh can be obtained by distributing these vertices in a sparse way over a larger area.

The refinement of an OLAM mesh occurs in Earth regions previously defined. The definition of the region to be refined begins with the choice of a specific geographic coordinate point. After this, all points included in the area formed by a radius surrounding that point will be refined.

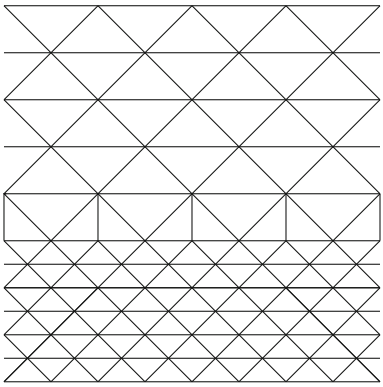

Fig. 3 Example of local mesh refinement transition from coarse to fine resolution

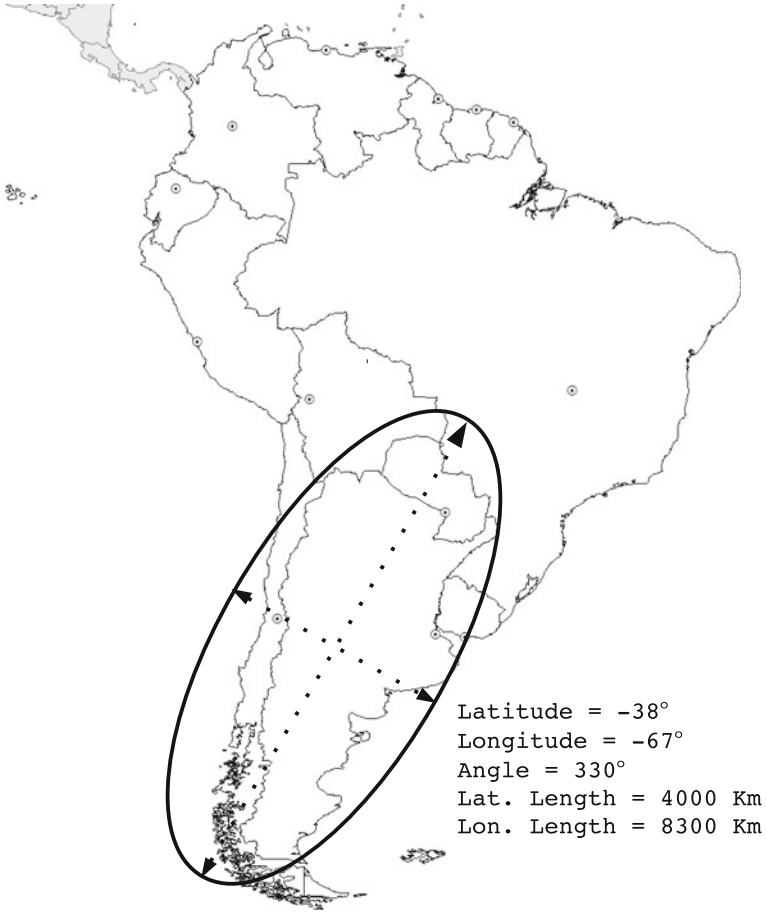

Fig. 4 Example of mesh refinement area definition to cover a specific region of the Earth surface

Thus, we can say that the refinement depends of a Cartesian coordinate, a radius of latitude, a radius of longitude and an angle of inclination of the ellipse formed by using the combination of these two radii. Figure 4 illustrates the distribution of each

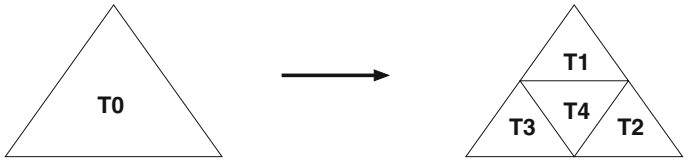

Fig. 5 Example of one level mesh refinement applied to a point

of these variables to define a region of refinement and how the choice of the angle allows you to rotate the ellipse in order to better cover a region of the physical world.

After the choice of the region, each triangle which barycenter belongs to the region is subdivided into four new triangles, as illustrated in Fig. 5. OLAM allows various levels of refinement that can be applied in different parts of the domain, that is, a given domain can be refined several times. Since the resolution of a level of refinement applied is always double in relation to the previous one, when x multi-level refinements are adopted, the final level of resolution will always be $initial\_resolution/2^x$.

4.3 Parallel Implementation

OLAM was developed and parallelized with message passing interface (MPI) [6]. 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 triangles (the mesh points of the domain) after the static mesh refinement to ensure a good load balance. Once defined the distribution of subdomains among the processes, each process discards the global mesh, and keeps in memory only its respective part of the global mesh.

5 Finer Mesh Resolution Execution

The OLAM mesh representation is made by decomposing the Earth surface into triangles, according to the requested resolution. The number of triangles for a specific resolution depends of Earth's circumference and is given by $20 \times (5050/R)$, where R is the resolution in Kilometers [14]. Table 1 presents the number of edges, vertices

Table 1 Number of vertices, edges and triangles mesh elements for different mesh resolutions

Resolution	Vertices	Edges	Triangles
100 km	25, 002	75, 000	50, 000
50 km	102, 012	306, 030	204, 020
10 km	2, 550, 252	7, 650, 750	5, 100, 500
5 km	10, 201, 002	30, 603, 000	20, 402, 000

and triangles for some mesh resolutions. In this table, we can see that the number of points increases one hundred times if the adopted resolution doubles.

For all the decomposed triangles, there is also a specific number of vertical atmosphere levels. This number depends of the mesh refinement level. For 200 km of resolution, we can use around 20 levels, for example, but for higher resolution, this number needs to be increased, in order to ensure the relationship between resolution and atmospheric size levels.

Physical data properties of the model are associated to edge and triangle elements of the mesh, and its specific vertical levels. Ignoring auxiliary data structures, and considering only physical proprieties, we need at least 30 data structures for a simple simulation.

If we also consider a long time simulation, in which we need many atmosphere simulation steps, each one representing a small real elapsed time, even using high performance architectures, the computational time is not acceptable.

For a parallel execution using at most 32 cores/processors, and execution parameters of 20 km of mesh resolution, 28 vertical levels, simulating only one day of atmospheric integrations, where each step represents 60 s of the real elapsed time, we need around 24 h of execution time. That is, for very simple simulation parameters, the simulation would be almost equal to the real time.

High speed execution of atmospheric models is fundamental to operational activities on weather forecast and climate prediction, due to execution time constraints—there is a pre-defined short time window to run a model. The model execution cannot begin before the arrival of input data, and cannot end after the due time established by user contracts. Experiences in international weather forecast centers point to a 2 h window to predict the behavior of the atmosphere in coming days. Operational models worldwide use the highest possible resolution that allows the model to run during the window on the available computer system.

Thus, the total execution time elapsed in the simulation is correlated to the number of elements that represent the domain. However, finer mesh refinement needs to be adopted only to cover local weather phenomena. In this context, to reduce the execution time, without loss of precision simulation, different mesh refinement levels can be used. The best solution to cover local phenomena is to adopt higher mesh resolution in a global Earth model when it is really necessary, using a run-time mesh refinement.

6 Online Mesh Refinement Implementation

The refinement of meshes at run-time takes into consideration that the domain points to be refined can be distributed into different processes. Thus, the implementation of this feature considers that each process must be able to identify whether its respective subdomain has a region to be refined.

Just as each process is responsible for setting its subdomain in the beginning of the code execution, each process realizes locally the identification of the area to be refined, since each point in the domain maintains a reference to geographical coordinates of the globe.

For the OMR we stop the execution and refine the distributed mesh on a specific point or Earth region according to a climatological condition. After this, the iterative execution proceeds normally.

All distributed processes know the mesh refinement that must be made in its specific sub-mesh. Each data structure of a mesh point has information about its position on the globe. Thus, each process knows which points must be refined. After the refinement, data structures of the new created points will be completed.

7 Performance Evaluation

In order to measure the performance impact of the OMR, we made several experiments. This section presents the simulation environment, execution parameters, and execution time measurements.

7.1 Execution Environment

All experimental measurements were obtained using a cluster formed by 6 Sun Fire X4600 stations, each one composed by 8 quad-core AMD Opteron 2.3 GHz processors and 128 GB of RAM memory distributed among the nodes of the cluster.

In all executions, we simulated the atmosphere for 24 h ahead. Each timestep simulates 60 s of the real time of the weather condition. The vertical atmosphere layer is divided into 28 layers. The number and the size of each one of this layers is chose according to the parameters adopted in large climatological simulation centers for its daily weather forecasts.

If a local mesh refinement is called, it is realized 8,700 km around a specific point of the Earth. The OMR occurred after 12 h of atmosphere simulation.

7.2 Online Mesh Refinement Execution Time Impact

A first test was made in order to analyze the impact of the OMR call on the total execution time. Figures 6, 7, 8 and 9 present the execution time results of an atmospheric integration, using a mesh with 100, 67, 50 and 40 km of horizontal mesh resolution, where an OMR occurs during the execution of the code. The graphics of these figures show the total execution time and the total time spent to call the OMR, using 1, 2, 4, 8, 16 and 32 processes.

Each column of the graphic represents the total execution time for a determined number of processes. This time includes the initialization step, iterative step before an OMR call, OMR execution and the iterative step after an OMR call. We can see that this time decreases when more processes are used.Consequently, there are performance gain.

The second measurement (scratched area) of each group of processes pres-ents the OMR execution time. The time duration of this step is approximately 130, 500, 570 and 800 s for the 100, 67, 50 and 40 km of mesh resolution cases, respectivelly.This time is a little more than the time spent with the initialization of the model, that is

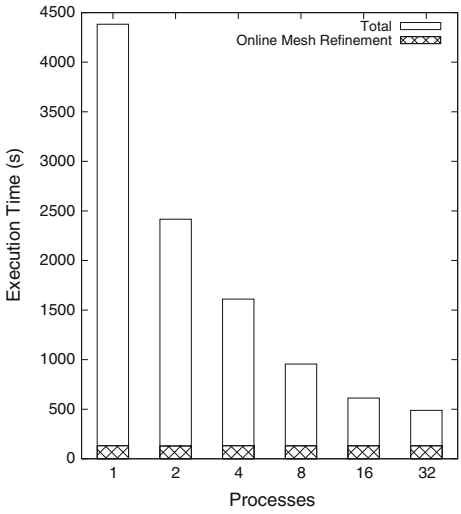

Fig. 6 Execution time using different number of processes for a 100 km mesh resolution with OMR call

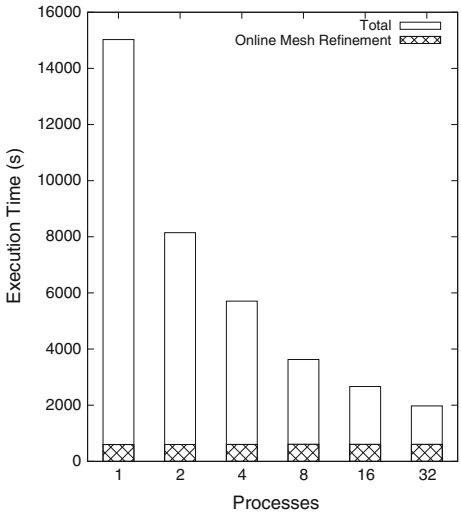

Fig. 7 Execution time using different number of processes for a 67 km mesh resolution with OMR call

115, 400, 415 and 550 s, respectivelly, for the four analyzed cases.The second time measurement includes all necessary procedures to interrupt the iterative step, to refine the mesh in each process and to reallocate variables.

The OMR has low impact on the total execution time. The time spent for this refinement is constant independently of the number of processes used in the atmospheric simulations. The relation between the time in the OMR call and the total execution time decreases if more high mesh resolutions are used.

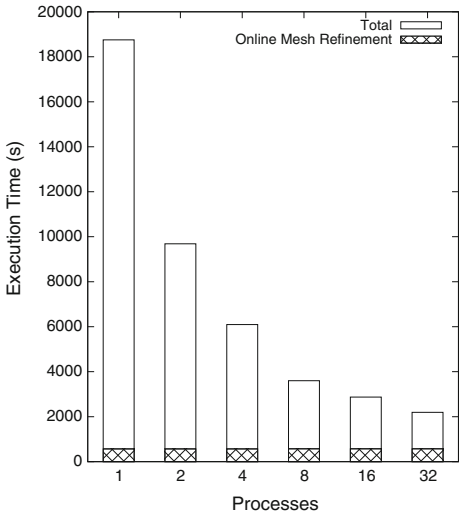

Fig. 8 Execution time using different number of processes for a 50 km mesh resolution with OMR call

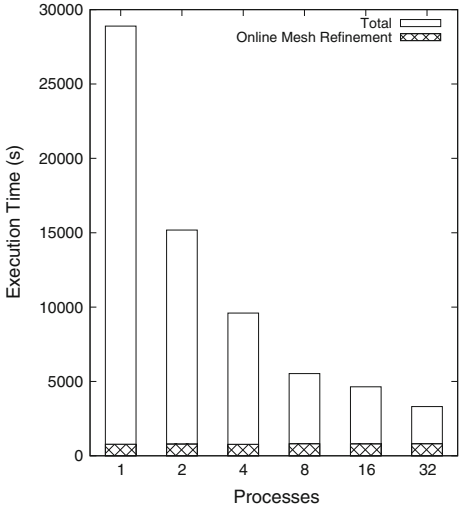

Fig. 9 Execution time using different number of processes for a 40 km mesh resolution with OMR call

7.3 Comparison between Static and Dynamic Mesh Refinement

The second test evaluates the execution time impact of a simulation using a runtime mesh refinement in relation to finer and larger global mesh refinements simulations. Figure 10 shows a comparison of the parallel execution time (in seconds) of 3 different configurations using 1, 2, 4, 8, 16 and 32 processes. The first and third columns show the total execution time using a global mesh resolution of 100 and 50 km respectively. The second column represents the total execution time for a 100 km grid resolution where an OMR occurs during the execution of the code.

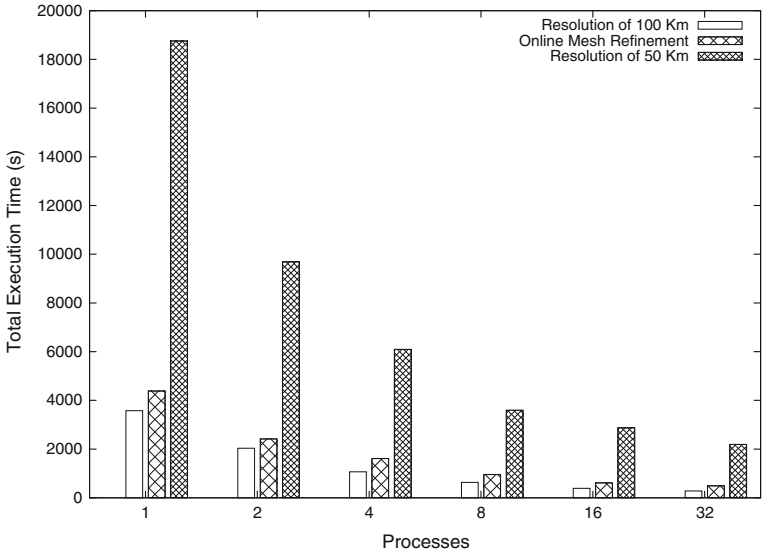

Fig. 10 Execution time using different number of processes for 100, 100 km with OMR and 50 km of mesh resolution

The results of Fig. 10 show that all configurations have a decrease of the execution time when a larger number of processes are used. The results demonstrate also that if we use a double resolution (50 km) instead of a large resolution (100 km), without run time mesh refinement call, we spend 5 to 8 times more execution time.

The execution time using OMR was always between the 100 km resolution and the 50 km resolution cases configuration. Thus, the evaluation of the implementation shows that it is efficient, since not all the surface of the Earth needs to be refined all the time. In fact, the total execution time increases a little in relation to the 100 km resolution case.

7.4 Speed up Evaluation of the Iterative Step of the Model

OLAM experimental simulations are made considering the speed up of the iterative step of the model in parallel simulations. In Figs. 11 and 12 are presented the speed up of the iterative step before and after an OMR call for a mesh with global resolution of 100 and 50 km, respectively. We range the number of MPI processes among 1, 2, 4, 8, 16 and 32.

In both cases, the white columns presents the speed up before an OMR call and the scratched columns the speed up after de OMR call. For both cases, the base to calcule the speed up was the execution time using a single process.

The use of more processes includes more performance in the iterative steps of the model for both mesh resolution cases. However, the speed up of the iterative step executed after the OMR call increases less than the speed up of the iterative

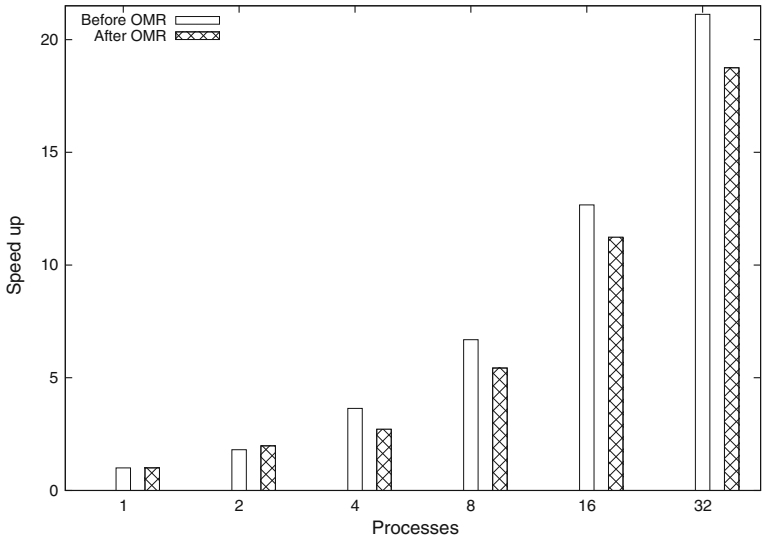

Fig. 11 Speed up comparison of the iterative step of the model before and after the OMR call for a global mesh resolution of 100 km

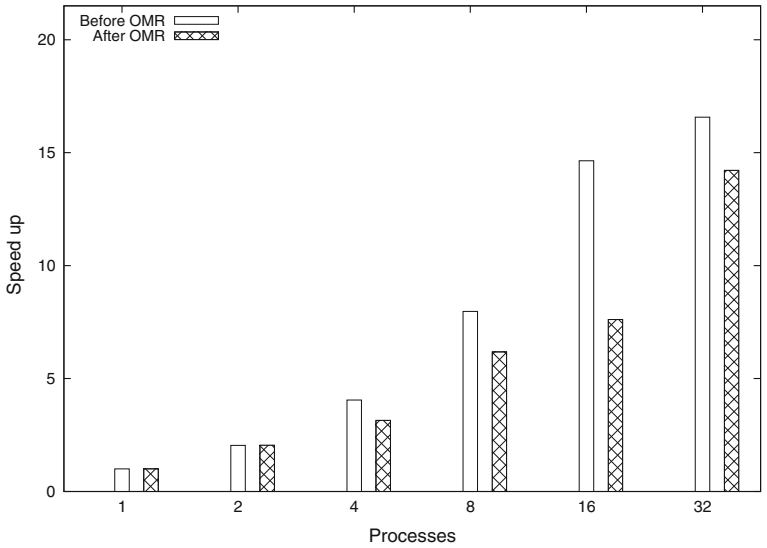

Fig. 12 Speed up comparison of the iterative step of the model before and after the OMR call for a global mesh resolution of 50 km

step executed before the OMR call. This occurs because of load unbalance among the processes. Why, and solutions to solve this issue, are discussed in the next section.

8 Improving Load Balance Distribution

The OMR approach leads to unbalanced load distribution after it is called, since the number of data elements increases in some processes. More data processing is required on the specific regions of the global domain that were refined. The refined points are not redistributed among all processes. Some processes may have new data elements to compute and others not.

8.1 Unbalanced Load Problem

Table 2 presents the number of decomposed elements for a domain with 100 km of mesh resolution divided in 8 processes before and after the OMR call.

In this table it is possible to see that the number of Vertices, Edges, and Triangles for the processes 2, 3, 6 and 7 increase after the mesh refinement execution. The localization of the increased points depends on the place of the Earth atmosphere where the mesh refinement occurs.

8.2 OpenMP Solution

In order to better distribute the load among the processes, we have added an OpenMP layer to the MPI program.

OpenMP is a parallel programming interface used to abstract multi-proces-sors architectures. The interface is also a good solution to explore parallelism in multi-core systems [3].OpenMP is also a good solution for climatological applications [10,11].

In this work, using OpenMP enables to benefit from thread-based concurrency, added to the MPI parallelism.Thus, each process divides the load among a specific number of threads.

Table 2 Unbalancing load after an OMR using 8 processes

Proc.	Before online refinement			After online refinement		
	Vertices	Edges	Triangles	Vertices	Edges	Triangles
0	3,408	9,873	6,468	3,408	9,873	9,873
1	3,394	9,925	6,534	3,394	9,925	6,534
2	3,412	9,952	6,543	5,667	16,549	10,857
3	3,439	10,041	6,605	5,933	17,362	11,404
4	3,421	9,959	6,541	3,421	9,959	6,541
5	3,432	10,042	6,613	3,432	10,042	6,613
6	3,451	10,070	6,622	6,427	18,491	12,067
7	3,452	10,131	6,682	5,952	17,556	11,607

8.3 Execution Time Using OpenMP Threads

The use of OpenMP threads was evaluated in some atmospheric simulations considering meshes with initial horizontal resolution of 100 km. Figures 13 and 14 present the speed up of the iterative step of the model before and after an OMR call, respectively. In the tests we compare 1, 2, 4 and 8 MPI processes for an atmospheric simulation using an initial horizontal mesh resolution of 100 km. We run 1, 2, 4 and 8 threads in each MPI process.

The results show that the use of threads OpenMP increases performance in the partial iterative steps before and after the OMR execution for all numbers of MPI processes evaluated.In all cases evaluated in Figs. 13 and 14, with exception of the last case (8 processes and 8 threads), only 1 workstation was utilized. If 8 MPI processes and 8 OpenMP threads are executed, then it is necessary to use 2 workstations. Thus, it is possible to run 8×8 threads in distinct cores, but this not increase the speed up more than the speed up obtained using other number of processes/threads, evaluated in only workstation.

Table 3 presents comparatively the speed up shown in Figs. 13 and 14.The first column indicates the kind of the partial iterative execution: 0–12 indicates the simulation before the OMR, 12–24 points the simulation after the OMR call. The second column shows the number of MPI processes used in each kind of partial iterative execution. The range from the third to the seventh column presents the speed up obtained using 1, 2, 4, and 8 OpenMP threads. The initial speed up are based on the sequential execution of each part of the iterative step. First line uses only OpenMP threads. The third column uses only MPI processes in the execution. The other mesurements combine MPI with OpenMP processes in the simulations.

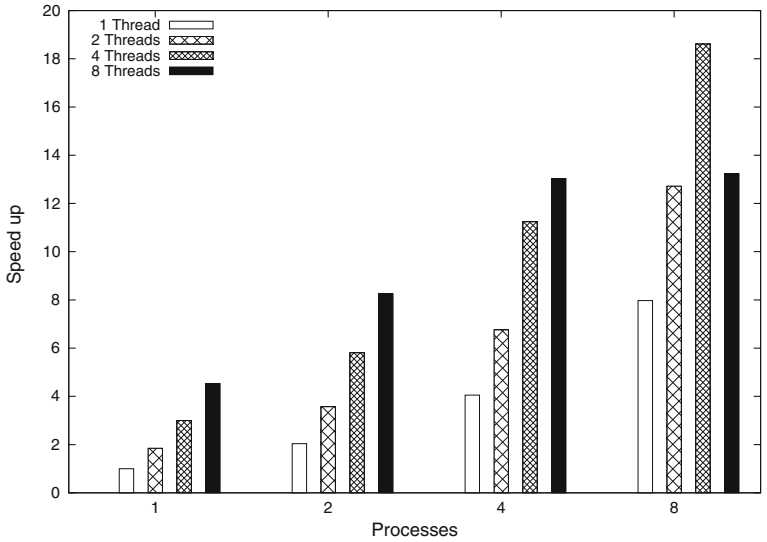

Fig. 13 Speed up of the iterative step executed before the OMR call using different number of OpenMP threads in a simulation with MPI processes

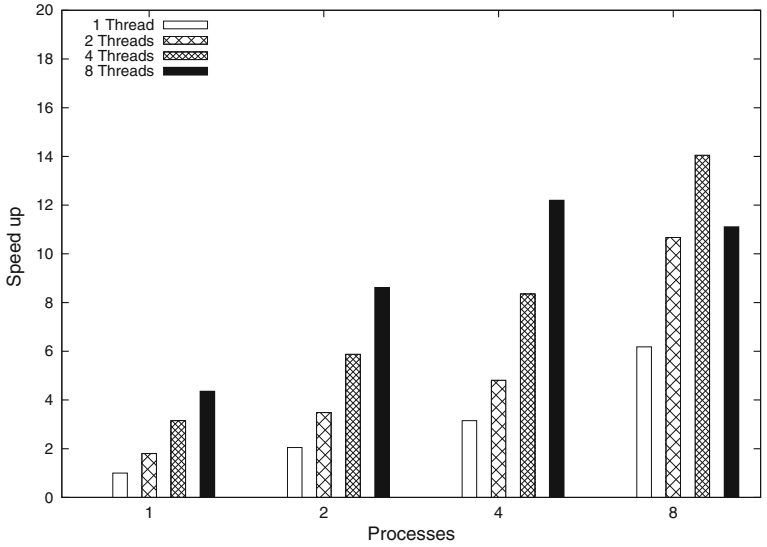

Fig. 14 Speed up of the iterative step executed after the OMR call using different number of OpenMP threads in a simulation with MPI processes

Table 3 Speed up for the 0–12 h and 12–24 h iterative execution steps

Step	Processors	1	2	4	8
0–12	1	1.00	1.85	3.00	4.53
12–24	1	1.00	1.80	3.15	4.36
0–12	2	2.04	3.57	5.81	8.26
12–24	2	2.05	3.48	5.88	8.62
0–12	4	4.05	6.76	11.24	9.70
12–24	4	3.15	4.81	8.35	8.86
0–12	8	7.97	8.16	10.69	13.24
12–24	8	6.18	8.68	10.55	11.11

The results presented in the table show that the second part of the iterative step has a speed up close to the first part in most of the cases. The results demonstrate that using more threads improves load balancing for the last part of the iterative step and, consequently, less total execution time.

9 Conclusions and Future Work

In this paper we have presented OMR as a way to improve the mesh resolution for climatological models without a significant increase in the execution time. This refinement scheme enables to refine the global mesh of a model during the execution of the code without rebooting the application. Mesh refinement at execution time is critical

for climatological models that will cover the impact of local phenomena, inputing more resolution only if it necessary.

We presented partial and comparative execution time in order to evaluate the overhead of the OMR. The partial measurement results show that there is a time spent with the refinement step. However, it pays because we do not need to run the code considering all Earth surface, with more resolution, all the time. Thus, high resolution is only adopted when special climatological conditions occur.

We also evaluated a mixed MPI/OpenMP parallel implementation. The code with OpenMP improves better parallel performance.

We are planning to include other load balancing resources in the OMR in future work. These solutions involve runtime creation of new processes through the use of primitives provided by the specification of the norm MPI2 [1,5] and by exploring Graphics Processing Units (GPU) architectures using Compute Unified Device Architecture (CUDA) [7].

Acknowledgments This work was supported by the Brazilian research foundation Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)—"National Counsel of Technological and Scientific Development".

References

1. Cera, M.C., Pezzi, G.P., Pilla, M., Maillard, N., Navaux, P.: Improving the dynamic creation of processes in MPI-2. In: Mohr, B., Träff, J., Worringen, J., Dongarra, J. (eds.) Recent Advances in Parallel Virtual Machine and Message Passing Interface. Lecture Notes in Computer Science, vol. 4192, pp. 247–255. Springer, Berlin (2006)
2. Chandra, R.: Parallel Programming in OpenMP. Morgan Kaufmann Publishers, San Francisco (2001)
3. Curtis-Maury, M., Ding, X., Antonopoulos, C.D., Nikolopoulos, D.S.: An evaluation of OpenMP on current and emerging multithreaded/multicore processors. In: Proceedings of the 2005 and 2006 International Conference on OpenMP Shared Memory Parallel Programming, IWOMP'05/IWOMP'06, pp. 133–144. Springer, Berlin (2008)
4. Fazenda, A.L., Demerval, S.M., Enari, E.H., Panetta, J., Rodrigues, L.F.: First Time User Guide (BRAMS Version 4.2) (2011)
5. Gropp, W., Ewing, L., Thakur, R.: Using MPI-2—Advanced Features of the Message-Passing Interface. The MIT Press, Cambridge (1999)
6. Gropp, W., Lusk, E., Doss, N., Skjellum, A.: High-Performance, Portable Implementation of the MPI Message Passing Interface Standard. Parallel Comput. **22**(6), 789–828 (1996)
7. Kirk, D.B., Hwu, W.W.M.: Programming Massively Parallel Processors: A Hands- on Approach. Morgan Kaufmann Publishers, San Francisco (2010)
8. MacNeice, P., Olson, K.M., Mobarry, C., de Fainchtein, R., Packer, C.: PARAMESH: A parallel adaptive mesh refinement community toolkit. Comput. Phys. Commun. **126**(3), 330–354 (2000)
9. Marshall, J., Adcroft, A., Hill, C., Perelman, L., Heisey, C.: A finite-volume incompressible Navier-Stokes model for studies of ocean on parallel computers. J. Geophys. Res. **102**(C3), 5753–5756 (1997)
10. Osthoff, C., Grunmann, P., Boito, F., Kassick, R., Pilla, L., Navaux, P., Schepke, C., Panetta, J., Maillard, N., Dias, P.L.S., Walko, R.: Improving performance on atmospheric models through a hybrid OpenMP/MPI implementation. In: The 9th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA 2011). IEEE Technical Committee on Scalable Computing, Busan, Korea (2011)
11. Osthoff, C., Schepke, C., Panetta, J., Grunmann, P.J., Dias, P.L.S., Kassick, R.V., Boito, F.Z., Navaux, P.O.A., Lopes, P.P., Fabricio, Souto, R.P.: OpenMP for accelerators performance evaluation on atmosphere model's application system. In: Proceedings of XXX Iberian-Latin-American Congress on Computational Methods in Engineering, 2011, Ouro Preto. Mecanica Computacional Vol. XXX, pp. -. Asociación Argentina de Mecánica Computacional (AMCA), Ouro Preto, Brazil (2011)

12. Plewa, T., Linde, T., Weirs, V.G.: Adaptive Mesh Refinement—Theory and Applications. Springer, Berlin (2003)
13. Schepke, C., Maillard, N., Osthoff, C., Dias, P.: Performance evaluation of an atmospheric simulation model on multi-core environments. In: Proceedings of Conferencia Latino Americana de Computación de Alto Rendimiento, pp. 330–332. Instituto de Informática/UFRGS, Gramado, RS, Brazil (2010)
14. Schepke, C., Maillard, N., Schneider, J., Heiss, H.U.: Online mesh refinement in parallel meteorological applications. In: Proceedings of Conferencia Latino Americana de Computación de Alto Rendimiento, Colima, Mexico (2011)
15. Schepke, C., Maillard, N., Schneider, J., Heiss, H.U.: 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). IEEE, Vitória, Espírito Santo (2011)
16. Schmidt, G.A., Ruedy, R., Hansen, J.E., Aleinov, I., Bell, N., Bauer, M., Bauer, S., Cairns, B., Canuto, V., Cheng, Y., et al.: Present-day atmospheric simulations using GISS model E: Comparison to in situ, satellite, and reanalysis data. J. Clim. **19**(2), 153 (2006)
17. Vasquez, T.: Weather Forecasting Red Book. Weather Graphics Technologies, Garland (2006)
18. Walko, R.L., Avissar, R.: The ocean-land-atmosphere model (OLAM). Part I: Shallow-water tests. Mon. Weather Rev. **136**(11), 4033–4044 (2008)
19. Washington, W.M., Parkinson, C.L.: An Introduction to Three Dimensional Climate Modeling, 2nd edn. University Science Books, Herndon (2005)