Improving Performance on Atmospheric Models through a Hybrid OpenMP/MPI Implementation

Carla Osthoff[1], *Pablo Grunmann*[1], *Francieli Boito*[2], *Rodrigo Kassick*[2], *Laercio Pilla*[2], *Philippe Navaux*[2], *Claudio Schepke*[2], *Jairo Panetta*[3], *Nicolas Maillard*[2], *Pedro L.Silva Dias*[1], *Robert Walko*[4]

[1] Laboratório Nacional de Computação Científica (LNCC), Brazil
osthoff,pablojg,pldsdias@lncc.br,

[2] Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS), Brazil
fzboito,rvkassick,llpilla,navaux,cschepke,nicolas@inf.ufrgs.br,

[3] Instituto Nacional de Pesquisas Espaciais (INPE), Brazil
jairo.panetta@cptec.inpe.br,

[4] University of Miami, USA
rwalko@rsmas.miami.edu

Abstract

This work shows how a Hybrid MPI/OpenMP implementation can improve the performance of the Ocean-Land-Atmosphere Model (OLAM) on a multi-core cluster environment, which is a typical HPC many small files workload application. Previous experiments have shown that the scalability of this application on clusters is limited by the performance of the output operations. We show that the Hybrid MPI/OpenMP version of OLAM decreases the number of output files, resulting in better performance for I/O operations. We also observe that the MPI version of OLAM performs better for unbalanced workloads and that further parallel optimizations should be included on the hybrid version in order to improve the parallel execution time of OLAM.

Keywords : *High Performance Computing, Multicore, Atmospheric Models,OLAM, I/O Performance Evaluation, Distributed System Application.* [1] [2]

1. Introduction

Numerical models have been used extensively in the last decades to understand and predict weather phenomena and the climate. In general, there are two kinds of models, differing on their domain: global (entire Earth) and regional (country, state, etc). Global models have spatial resolution of about 0.2 to 1.5 degrees of latitude and therefore cannot represent very well the scale of regional weather phenomena. Their main limitation is computing power. On the other hand , regional models have higher resolution but are restricted to limited area domains. Forecasting on limited domain demands the knowledge of future atmospheric conditions at domain borders. Therefore, limited area models require previous execution of global models.

A novel, interesting approach was recently developed at Duke University. The main feature of this model called Ocean-Land Atmosphere Model (OLAM) [13] 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 scale and the local scale phenomena, as well as bi-directional interactions between scales.

Moreover, recent workload studies shows that many HPC storage systems are used to store many small files in addition to the large ones. Further investigation finds that these files come from a number of sources, not just one misbehaving application [(Carns et. al. (2009))]. Several scientific domains such as climatology, astronomy, and biology generate data sets that are most conveniently stored and organized on a file system as independent files. In this scenario, it is imperative to investigate the scale of programs for such workload applications on multilevel parallelism environment.

In previous works, [6] and [10] the performance of the Atmospheric Model Simulation OLAM was evaluated on a multi-core cluster environment and it was demonstrated that the scalability of the application on the system is limited

[1]Supported by CNPq.

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

by the performance of the output operations. In [7], it was shown that we can improve the application speed-up curve up to a certain number of cores allocation.

This work shows that a Hybrid MPI/OpenMP implementation can improve the Atmospheric Model Simulation OLAM (Ocean-Land-Atmosphere Model) application performance on a multi-core cluster environment. We observed that the Hybrid MPI/OpenMP version of OLAM decreases the number of output files, resulting in better performance for I/O operations. We also notice that the MPI version of OLAM performs better for unbalanced workload and that we need to include further parallel optimizations in the Hybrid MPI/OpenMP version of OLAM in order to improve the parallel execution time of the OLAM algorithm.

The remainder of this paper is organized as follows. Section 2 presents Ocean-Land-Atmosphere Model algorithm and OLAM Simulation Analysis. The performance evaluation, experimental results and experimental analysis are presented in section 3. Section 4 presents related work. The last section presents conclusions and future works.

2 Ocean-Land-Atmosphere Model

OLAM was developed to extend features of the Regional Atmospheric Modeling System (RAMS) to the global domain [9]. OLAM uses many functions of RAMS, including physical parameterizations, data assimilation, initialization methods, logic and coding structure and I/O formats [13]. OLAM introduces a new dynamic core based on a global geodesic grid with triangular mesh cells. It also uses a finite volume discretization of the full compressible Navier Stokes equations. Local refinement can be specified to cover specific geographic areas with more resolution. Recursion may be applied to a local refinement. 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.

The 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 for acoustic terms. The finite volumes are defined horizontally by the global triangular-cell grid mesh and subdivided vertically through the height of the atmosphere forming vertically-stacked prisms of triangular bases. The global domain greatly expands the range of atmospheric systems and scale interactions that can be represented in the model, which was the primary motivation for developing OLAM.

OLAM was developed in FORTRAN 90 and recently parallelized with Message Passing Interface (MPI) under the Single Program Multiple Data (SPMD) model.

The geodesic grid offers important advantages over the commonly used latitude-longitude grid. It allows mesh size to be approximately uniform over the globe, avoiding singularities and grid cells of very high aspect ratio near the poles. OLAM's grid construction begins from an icosahedron inscribed into an spherical earth, as is the case for most other atmospheric models that use geodesic grids.

An 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 $\pm tan^{-1}(1/2)$.

Uniform subdivision of each icosahedral triangle into $N \times N$ smaller triangles, where N is the number of edge 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 projected radially outward to the sphere to form geodesics.

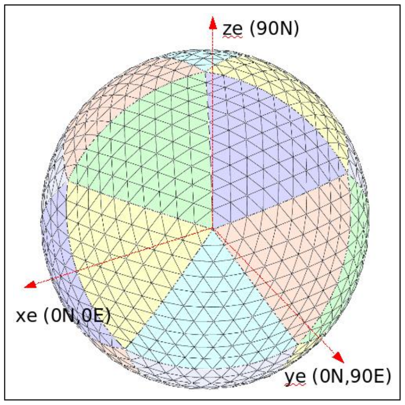

Figure 1. OLAM subdivided icosahedral mesh and cartesian coordinate system with origin at Earth center [13]

2.1 OLAM Typical simulation Analysis

Our first case study replicates a typical global climate simulation with a 200 km horizontal resolution, requiring the subdivision of each icosahedron edge in 25 parts. So, the distance between grid points on the globe was around 200 Km. The represented atmosphere was vertically-divided in 28 horizontal layers. We simulate 24 hours integration of the equations of atmospheric dynamics without any additional physical calculation (such as moisture and radiative 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.

An OLAM typical simulation requires reading 0.5GB 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). Each client reads the whole input file, thus increasing the initialization time with larger executions. After this phase, 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 do hide the cost of message transmission in the time spent processing messages previously received.

After executing a number of timesteps, the variables representing the atmosphere are written to a history file. During this phase, each MPI process opens its own history file for that superstep, writes the atmospheric state and closes the history file. Each client executes these three operations independently, since there is no collective I/O implemented in OLAM. On the other hand, there is an implicit barrier soon after generating the history files due to the aforementioned communication. These history files are considered of small size for the standards of scientific applications: each file size ranges from 100 to 600 KB, depending on the grid definition and number of MPI processes employed.

For the first case study problem size, OLAM application typically writes a 2 Mb output history file, 200 Kb output plot files for each core and 500 Kb output results files for each core, at the end of the simulation. Therefore, as we increase the number of cores, we increase the number of independent output files. We divide OLAM algorithm in three parts: the parameter initialization part, the atmospheric time state calculation part and the output write results part. Figure 2 presents the OLAM algorithm fluxogram.

In order to avoid the overhead imposed by the creation of a large number of files, an alternative version of OLAM was recently developed with MPI and OpenMP. In this version, each OLAM MPI process creates OpenMP threads at the start of the timestep and destroy them after the results are output. Therefore, this version, in this paper called Hybrid MPI/OpenMP version of OLAM, uses a different level of parallelism, generating less files. This happens because it maintains the same parallelism degree, but with a smaller number of MPI processes (each of them with a number of threads). As each output file correspond to one MPI Rank,

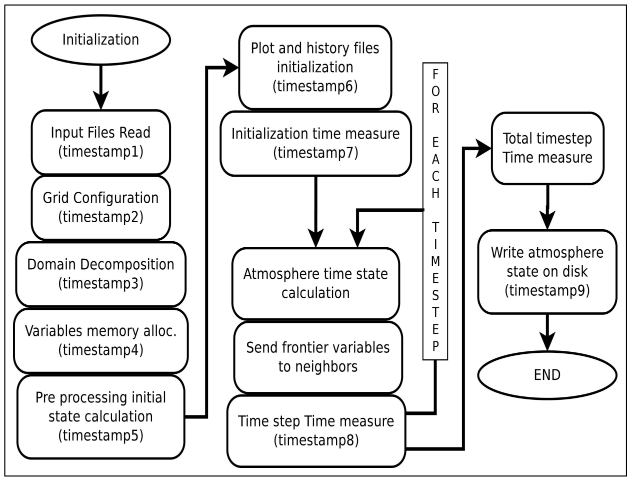
Initialization
Input Files Read (timestamp1)
Grid Configuration (timestamp2)
Domain Decomposition (timestamp3)
Variables memory alloc. (timestamp4)
Pre processing initial state calculation (timestamp5)
Plot and history files initialization (timestamp6)
Initialization time measure (timestamp7)
Atmosphere time state calculation
Send frontier variables to neighbors
Time step Time measure (timestamp8)
FOR EACH TIMESTEP
Total timestep Time measure
Write atmosphere state on disk (timestamp9)
END

Figure 2. OLAM algorithm fluxogram

the total number of generated files decreases compared to the previous version of OLAM. The OpenMP parallelism is applied to an outer DO LOOP over horizontal locations (grid columns) during each timestep calculation. OpenMP parallelism is applied to routines which are responsible for 70% of timestep phase CPU execution time and 75% of total cache miss [7] for MPI version of OLAM running 8 MPI process in 8 cores of one multi-core node.

3 Performance Evaluation

3.1 Simulation Environment

The performance measurements were made on a multi-core SGI Altix-XE 340 cluster platform (denoted Altix), located at the National Laboratory of Scientific Computing (LNCC). Altix-XE is composed of 30 dual node Intel Xeon E5520 2.27GHz Quad Core with 128Kb L1 cache, 1024KB L2 cache, 8192KB L3 cache and 24GB of main memory in each node, interconnected by an *Infiniband* network and it is using MPICH version 2-1.4.1 and Vtune Performance Analyzer version 9.1. The results presented here are the arithmetic average of four executions. The Hyperthread system is active and Turbo-boost increases processor speed when only one core is active. The following sections present and discuss our results.

The experiments evaluate the performance for MPI version of OLAM version 3.3 and for a Hybrid MPI/OpenMP version of OLAM version 3.3. Both parallel versions are developed by Robert Walko [13]. MPI version of OLAM starts one MPI process on the cores of the nodes cluster platform. Hybrid MPI/OpenMP version of OLAM starts one MPI process on each node of the cluster, and each MPI process starts OpenMP threads on the cores of the nodes cluster platform.

3.2 Experimental Results and Analysis

We present three experimental analysis in three distinct OLAM workload configurations. In the first experiment we performed a 200km OLAM configuration analysis, a typical horizontal resolution most commonly used for climate simulation.

The second experiment performed a 40km OLAM configuration. This is typical resolution of global forecast uses. This experiment decreases 5 times the horizontal distance between points in the globe, hence more points are necessary to cover the same area while the other parameters remain the same as in the 200km configuration. Since it is necessary 25 times the number of points as before to cover the same area at 5 times shorter space intervals, the number of calculations per timestep is now increased 25 times with respect to the previous experiment. Furthermore, it implies a 20-fold increase in the memory workload, thus increasing the the proportion of computing time with respect to data transfers.

The third experiment performs a 40km OLAM configuration analysis with additional physics calculation (including moisture, radiative and other processes of a realistic atmosphere simulation). This simulation generates additional unbalanced workload to be calculated over the globe. This is a typical unbalanced simulation workload due to the spacial heterogeneity introduced by specialized physical parameterizations when and where certain conditions develop.

3.2.1 200km OLAM Experiment

Figure 3 presents ideal and measured speed-up from 1 to 80 cores for 200km OLAM running with MPI version and Hybrid MPI/OpenMP version of OLAM. We observe that, for this configuration, the performance for both MPI-only and Hybrid MPI/OpenMP version of OLAM keeps similar up to 16 cores, as we increase the number of cores, Hybrid MPI/OpenMP version of OLAM performs better than the MPI-only version.

Previous works, [6] and [10], evaluate the performance of the Atmospheric Model OLAM (Ocean-Land-Atmosphere Model) on a multi-core cluster environment and demonstrate that the scalability of the system is limited by output operations performance. The work also shows that one of the reasons for the overhead in these operations is the low aggregated I/O throughput capacity of the multi-core system. Therefore, OLAM suffers significantly due to file creation and small write requests.

Recent work [2] evaluates that the new version of Hybrid MPI/OpenMP version of OLAM, generates less files than the MPI-only version, reducing the number of files up to 1/8 of as before on each node that resulting in up to 20 times better performance for I/O operations.

Therefore, in the figure 3 the superior performance of Hybrid MPI/OpenMP version of OLAM is mostly due to the reduction in the number of files and consequent improved performance for I/O operations.

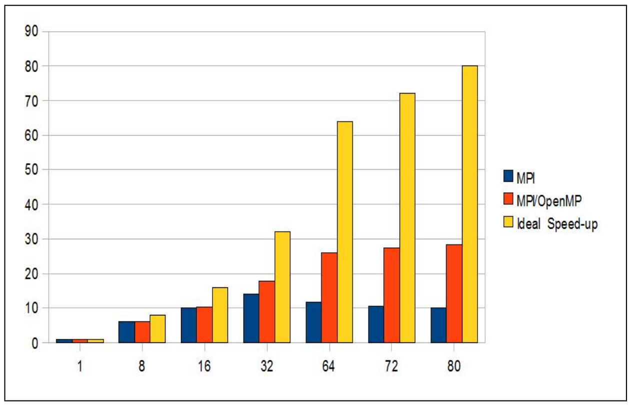

Figure 3. OLAM 200km ideal and measured MPI and MPI/OpenMP version speed-up.

3.2.2 40km OLAM Experiment

Figure 4 presents measured speed-up from 1 to 80 cores for 40km MPI version and Hybrid MPI/OpenMP version of OLAM compared to ideal speed-up. We observe that, for this configuration, MPI-only version performs better than Hybrid MPI/OpenMP version of OLAM up to 32 cores, but, as we increase the number of cores, Hybrid MPI/OpenMP version performs better than the MPI-only version of OLAM. This test shows that as we increase the dependency relationship between computing time and data transfers, output operations overhead decreases overall system performance impact.

In order to explain why MPI version of OLAM performs better than Hybrid MPI/OpenMP version of OLAM at the lower number of cores (up to 32) limit, we inserted VTUNE Analyser performance instrumentation in both codes.

We observe that, for this configuration, as we increase the number of processes in one node from the MPI version of OLAM, memory allocation grows from 2,0GB for one process up to 5GB for 8 processes. As we increase the number of threads in one node from the Hybrid MPI/OpenMP version of OLAM, the memory allocation keeps almost the same size (2,0GB), due to the shared memory data allocation. These results confirm that OpenMP global memory implementation improve OLAM memory data allocation on a multi-core system.

On the other hand, Hybrid MPI/OpenMP version of OLAM directives are inserted into a routine that was found to be responsible for up to 70% of the cpu time from the timestep phase execution time [6] and [10] and up to 75% of cache misses for MPI version of OLAM running 8 MPI process in 8 cores of one multi-core node. We observe that, after this parallelization, the cpu time consumption was greatly reduced for this routine (from 70% to about 28% of the total now). After these modifications, we can see other two routines are now taking about 44% of the cpu time.

Also, for this configuration, we observe that only 28% of the code is running in parallel and the rest is still running on a single core. Which explains the poor performance of the Hybrid MPI/OpenMP version of OLAM code running with 8 threads in one node. Therefore, as we increase the number of nodes in the cluster, Hybrid OLAM MPI/OpenMP generates less I/O overhead than the MPI version of OLAM, improving overall system performance.

We conclude that Hybrid MPI/OpenMP version of OLAM needs further optimizations in order to improve OLAM algorithm parallel execution time.

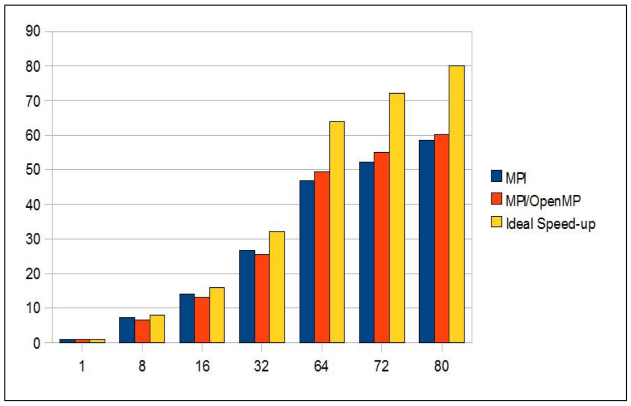

Figure 4. OLAM 40km ideal and measured MPI and MPI/OpenMP speed-up.

3.2.3 40km OLAM with additional Physics calculation Experiment

This experiment performs a 40km OLAM configuration analysis with additional physics calculation, which generates additional unbalanced workload to be calculated over the globe.

Figure 5 presents ideal and measured speed-up from 1 to 80 cores for 40km MPI version of OLAM with physics and Hybrid MPI/OpenMP version of OLAM compared to ideal speed-up. We observe that, for this configuration, MPI version of OLAM performs even better than the Hybrid MPI/OpenMP version of OLAM. We also observe that, as we increase the number of nodes, Hybrid MPI/OpenMP version of OLAM advances more in performance than MPI-only version decreasing the gap due to the Hybrid MPI/OpenMP version of OLAM output performance benefits.

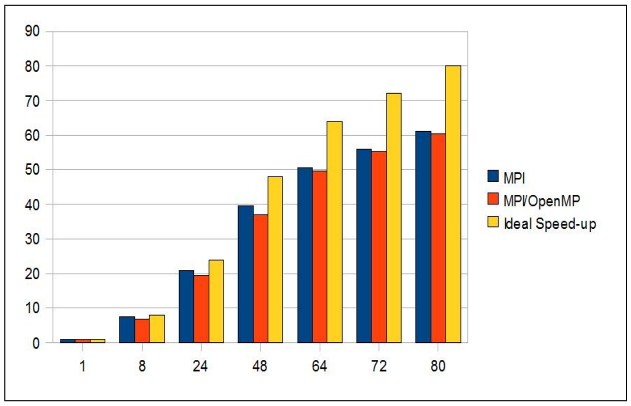

Figure 5. 40km with physics ideal and measured MPI and MPI/OpenMP speed-up.

This experiment shows that as we increase the unbalanced data workload, the performance impact is greater for the Hybrid MPI/OpenMP version of OLAM. The reason is that unbalanced workload decreases the relative portion of parallel execution in the OLAM algorithm.

4 Related Works

Parallel applications scalability is the focus of several papers in the literature. In [4] the authors execute the high resolution WRF weather forecast code over 15k cores and conclude that one of the greatest bottlenecks is data storage. This same code was used in [11] to evaluate file system caching and pre-fetching optimizations to many-core concurrent I/O, achieving improvements of 100% I/O contention on multi-core systems is also a known issue in the literature and a few strategies to mitigate the performance loss can be found. In [8], the authors improve the performance of multiple concurrent I/O streams on multi-core nodes using data aggregation, forcing contiguous access on the storage nodes. A similar strategy is employed in IOFSL [1], an initiative from the $ParallelVirtualFileSystem$ (PVFS) team. Optimizations to increase OLAM scalability are being currently studied. [10] presents an evaluation of OLAM MPI with $VtuneAnalyzer$ and identifies a large amount of cache misses when using 8 cores. More recently, [2] tested OLAM on the PVFS file system in order to point the I/O characteristics of the application for 200km horizontal resolution most commonly used for climate simulation. Our work also tested OLAM in this resolution but on a typical NFS file system cluster. Besides, we also tested higher resolution workloads more typical of global forecast uses and for typical unbalanced simulation workload due to the spacial heterogeneity introduced by specialized physical parameterizations when and where certain conditions develop.

5 Conclusions and Future Works

This work shows that a Hybrid MPI/OpenMP implementation can improve the performance of an application that requires many small files on a multi-core NFS File System cluster environment. We observe that as we increase the number of cluster nodes in the system, Hybrid MPI/OpenMP version of OLAM performs better than the MPI version of OLAM. The main reason is that the Hybrid MPI/OpenMP version of OLAM decreases the number of output files, resulting in a better performance for I/O operations.

We also observe that, as we increase the proportion of computing time with respect to data transfers, there is a decrease in the impact of the output operations overhead over the overall system performance. We also confirm that OpenMP global memory advantages improves the application memory data allocation in a multi-core system. But, on the other hand, we observe that there is a decrease in the parallelization execution time, for instance, OLAM 40km resolution runs only 28% of the code in parallel and the rest is still running on a single core. We conclude that we need to include further parallel optimizations in the Hybrid MPI/OpenMP version of OLAM in order to improve the proportion of computing time with respect to data transfer.

Finally, we observe that when running with additional unbalanced workload, the MPI version performs even better than the Hybrid MPI/OpenMP version for our application, due to the further increase in the serial part of the code.

For future works we envision several directions. In order to include further parallel optimizations in Hybrid MPI/OpenMP version of OLAM we plan to add further OpenMP parallelization into the OLAM code and to investigate memory data allocation algorithms in order to decrease unbalanced data workload.

In order to include further I/O operations performance optimizations we also intend to evaluate the parallel I/O library from [3] aiming to reduce OLAM initialization file creation overhead and the work of [5] aiming to improve output write operations performance.

References

[1] Ali, N., Carns, P., Iskra, K., Kimpe, D., Lang, S., Latham, R., Ross, R., Ward, L., and Sadayappan, P. *Scalable I/O Forwarding Framework for High-performance Computing Systems.* ,pages 1 10.

[2] Boito, F., Kassick, R.,Pilla, L.,Dias,P.,Panetta,J. *I/O Performance of a Large Atmospheric Model using PVFS.* LNCC Reaserch Repport , 2011, url:www.lncc.br

[(Carns et. al. (2009))] Carns, P., et all. *Small-File Access In Parallel File Systems.* Parallel and Distributed Processing, IPDPS 2009.1530-2075 pages1 -11.,2009.

[3] Marshall, J., Adcroft, A., Hill, C., Perelman, L., Heisey, C. *Scalable massively parallel I/O to task-local files.* Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, http://doi.acm.org/10.1145/1654059.1654077, 2009.

[4] Michalakes, J., Hacker, J., Loft, R., McCracken, M. O., Snavely, A., Wright, N. J., Spelce, T., Gorda, B., and Walkup, R. *WRF Nature Run.* Journal of Physics: Conference Series, 125(1):012022.

[5] Nawab, A ., et all. *Scalable I/O Forwarding Framework for High-Performance Computing Systems.* Cluster Computing and Workshops, CLUSTER09. IEEE International Conferenc, pages 1-10, 2009.

[6] Osthoff,C.,Schepke,C., Panetta, J., Grunmann, P., Silva Dias,P. L., Maillard,N., Navaux,P.,Lopes,P.P., *I/O Performance Evaluation on Multicore Clusters with Atmospheric Model Environment* Proceedings of 22nd International Symposium on Computer Architecture and High Performance Computing-WAMMCA-2010.

[7] Osthoff,C.,Schepke,C., Panetta, J., Grunmann, P., Silva Dias,P. L., Kassick,R., Boito,F., Navaux,P. *Improving Core Selection on a Multicore Cluster to Increase the Scalability of an Atmospheric Model Simulation* Proceedings of XXXI Iberian-Latin-American Congress on Computational Methods in Engineering- CILAMCE-2010.

[8] Ohta, K., Matsuba, H. And Ishikawa, Y. *Improving ParallelWrite by Node-Level Request Scheduling* In CCGRID 09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 196203,Washington, DC, USA, 2009. IEEE Computer Society

[9] Pielke, R.A., et al. *A Comprehensive Meteorological Modeling System - RAMS.* Meteorology and Atmospheric Physics. 49(1), pages 69-91, 1992.

[10] Schepke,C.,Maillard,N., Osthoff,C.,Dias,P.,Pannetta,J. *Performance Evaluation of an Atmospheric Simulation Model on Multi-Core Environments* Proceedings of the Latin American Conference on High Performance Computing (CLCAR), pages 330-332.

[11] Seelam, S., Chung, I.-H., Bauer, J., Yu, H., and Wen, H.-F. *Application level I/O caching on Blue Gene/P systems.* ,pages 1 8.

[12] Sun, X., Byna, S.,Holmgren,D. *Modeling Data Access Contention In Multicore Architectures.* Parallel and Distributed Systems (ICPADS), 2009 15th International Conference, pages 213-219, 2009

[13] Walko, R.L., Avissar, R. *OLAM: Ocean-Land-Atmosphere Model - Model Input Parameters - Version 3.0.* Tech. Rep., Duke University , November, 2008.