Asociación Argentina de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 3143-3153 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

IMPROVING CORE SELECTION ON A MULTICORE CLUSTER TO INCREASE THE SCALABILITY OF AN ATMOSPHERIC MODEL SIMULATION

Carla Osthoff [a], Claudio Schepke [b], Jairo Panetta [c], Pablo Grunmann [a], Pedro L. Silva Dias [a], Rodrigo Kassick [b], Francieli Boito [b] and Philippe Navaux [b]

[a]*Laboratório Nacional de Computação Científica (LNCC), Caixa Postal 25.651 075, Petrópolis, RJ, Brazil, (osthoff, pldsdias, pablojg)@lncc.br, http://www.lncc.br*

[b] *Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS), Caixa Postal 15.064 - 91.501-970, Porto Alegre, RS, Brazil, (cschepke, rodrigovirote.kassick, fzboito, navaux)@inf.ufrgs.br, http://www.inf.ufrgs.br*

[c]*Instituto Nacional de Pesquisas Espaciais (INPE), Caixa Postal 12630-000, Cachoeira Paulista, SP, Brazil, jairo.panetta@cptec.inpe.br, http://www.cptec.inpe.br*

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

Keywords: High Performance Computing, multicore, Atmospheric Model Simulation.

Abstract. This work shows that the we can improve the application speed-up curve up to a maximum number of cores allocation according to two Ocean-Land-Atmosphere Model (OLAM) workload parameters on a multicore cluster environment. Previous experiments have shown that the scalability of the system is limited by output operations performance. We show that one of the reasons for such output operations overhead is the low aggregated I/O throughput capacity of the multicore system. A higher capacity would be desirable to achieve better concurrent access rates. We show that, for a given atmospheric model application configuration, there is an maximum number os cores allocation that improves application speed-up curve.

1 INTRODUCTION

While multicore processors offer computing power to improve the performance of applications by running multiple threads concurrently, it is the developers responsability to efficiently employ them. In order to achieve the best performance of these processors, there are two popular models: Thread (or Task) Level Parallelism (TLP) and Data Level Parallelism (DLP). Both models bring several design issues, data access latency being one of great importance.

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. This will increase the impact of multiple levels of parallelism on clusters.

In previous works, (Osthoff et. all.(2010) and (Schepke et. al. (2010), we evaluated the performance of the Ocean-Land-Atmosphere Model (OLAM) on a multicore cluster environment and demonstrated that the scalability of the system is limited by output operations performance. We have also shown that one of the reasons for the overhead in these operations is the low aggregated I/O throughput capacity of the multicore system. A higher capacity would be desirable to achieve better concurrent access rates

Therefore, we assume that we can improve the application speed-up curve up to a maximum number of cores allocation for a given OLAM configuration that allows maximal use of the cores of a multicore platform without causing output operations overhead. In order to validate our assumptions, this work evaluates the maximum number of cores according to two typical OLAM workload configurations.

The contribution of this work is to show that for given OLAM configuration there is an maximum number of cores that improves the scalability of OLAM runs on a multicore cluster platform.

The remainder of this paper is organized as follows. In the next section we present related work. Section 3 presents atmospheric model performance problem and OLAM algorithm. The performance evaluation, experimental results and experimental analysis are presented in section 4. The last section presents conclusions and future works.

2 RELATED WORK

Data access latency has been a problem even on single-core systems, as processors are much faster than memory. With the emergence of multicore processors, data access bandwidth becomes a severe problem, due to concurrent access to shared resources in the memory hierarchy (e.g. caches). When multiple cores are processing different sets of data, the shared resource becomes a performance bottleneck, if the bandwidth is not high enough to support the multiple cores. This has been already experienced in currently available processors (Shalf et. al. (2007). In order to better use multicore resources, the work of (Sun et. al. (2009) introduces simple analytical models for predicting the occurrence of data access contention and provides a guideline for choosing the optimum number of cores to avoid data access contention while running an application.

Another work (Datta et. al. (2009) presents a set of effective optimizations and an auto-tuning environment that searches over the optimization space to minimize runtime. The strategy allows performance portability over multiple architectures, due to auto-tuning.

3 ATMOSPHERIC MODEL PERFORMANCE PROBLEM

High speed implementation 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 the model. Model execution cannot begin before input data arrives, and cannot end after the due time established by user contracts. Experience in international weather forecast centers points to a two-hour window to predict the behavior of the atmosphere in coming days.

In general, atmospheric and environmental models comprise a set of Partial Differential Equations which include, among other features, the representation of transport phenomena (hyperbolic equations). Their numerical solution involves time and space discretization subject to the Courant Friedrichs Lewy condition (CFL condition) for stability which imposes a certain proportionality between the time and space resolutions (inverse of the distance between points in the domain mesh). For a 1-dimensional mesh, the number of computing points, n, is given by L/d, where "L" is the size of the domain to be solved and "d" is the distance between points over this domain. In our case, the mesh is 4-dimensional (3 for space and 1 for time), therefore the computational cost is of $O(n^4)$, where n is the number of latitude (or longitude) grid points in the geographical domain of the model, if the number of vertical points also increases with n. The spacing between consecutive points (called "resolution") strongly influences the accuracy of results.

Operational models worldwide use the highest possible resolution that allow the model to run at the established time window in the available computer system. New computer systems are selected for their ability to run the model at even higher resolution during the available time window. Given these limitations, the impact of multiple levels of parallelism and multicore architectures in the execution time of operational models is indispensable research.

Numerical models have been used extensively in the last decades to understand and predict the climate and weather phenomena. 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. The central limitation is computing power. Regional models, on the other hand, have higher resolution but are restricted to limited area domains. Forecasting the future on a 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) is to provide a global grid that can be locally refined, forming a single grid. That allows simultaneous representation (and forecasting) of both the global scale and the local scale phenomena, as well as bi-directional iterations among scales (Walko et. al. (2008).

3.1 Ocean-Land-Atmosphere Model

OLAM was developed to extend features of the Regional Atmospheric Modeling System (RAMS) to the global domain (Pielke et. all.(1992). OLAM uses many functions of RAMS, including physical parameterizations, data assimilation, initialization methods, logic and coding structure and I/O formats (Walko et. al. (2008). 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.

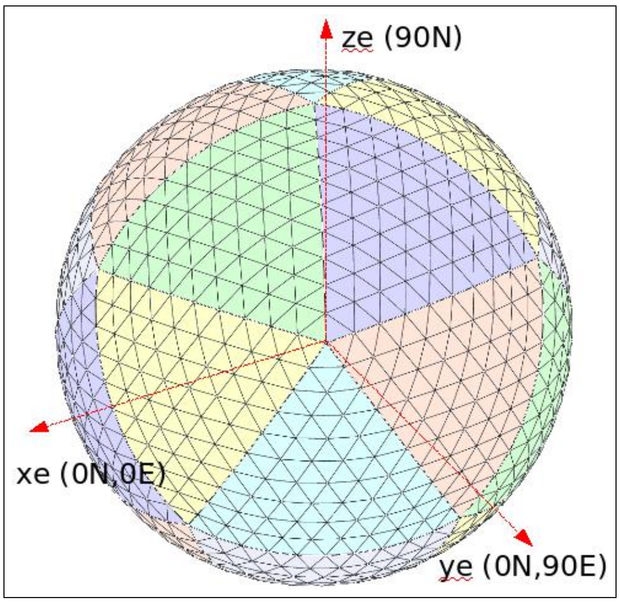

Figure 1: OLAM subdivided icosahedral mesh and cartesian coordinate system with origin at Earth center (Walko et. al. (2008)

The model consists essentially of a global triangular-cell grid mesh with local refinement capability, the full compressible nonhydrostatic Navier-Stokes equations, a finite volume formulation of conservation laws for mass, momentum, and potential temperature, and numerical operators that include time splitting for acoustic terms. 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.

3.2 OLAM Global Grid Structure

Figure 1 shows an example of the OLAM subdivided icosahedral mesh and cartesian coordinate system with origin at Earth center. The projection causes most triangles to deviate from equilateral shape, which is impossible to avoid (Walko et. al. (2008). 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.

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. 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.

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.

3.3 OLAM Typical simulation Analysis

Our first case study replicates a typical global forecast with 200 km horizontal resolution, requiring the subdivision of each icosahedron edge in 25 parts". So, the distance between points on the globe was around 200 Km. 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 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 process opens the 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 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. Finally, we inserted timestamps barriers on selected points of OLAM source (a few module boundaries) in order to correctly assing partial execution times to OLAM main modules.

4 PERFORMANCE EVALUATION

The performance measurements were made on a multicore SUN cluster platform (denoted SunHPC), located at the National Laboratory of Scientific Computing (LNCC) is composed of 72 dual node Intel Xeon E5440 Quad-Cores with 4 MB of cache and 16 GB of RAM memory in each node, interconnected by an *Infiniband* network and is using MPICH version 2-1.2.p1 and Vtune Performance Analyzer version 9.1. The following sections present and discuss our

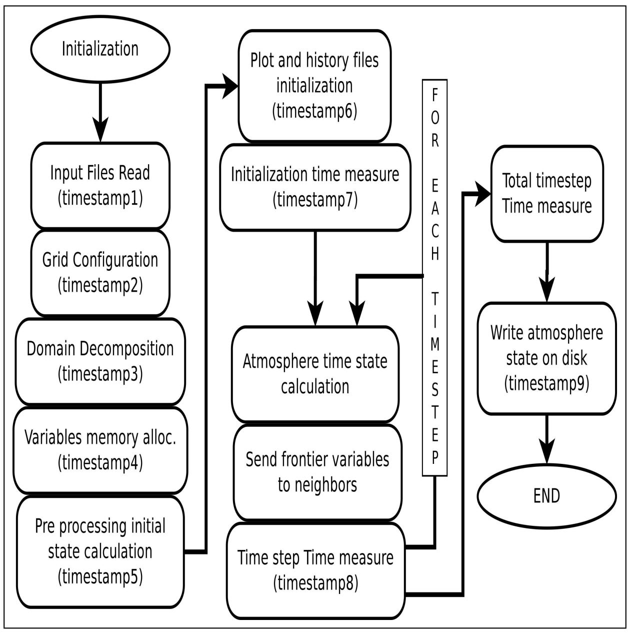
Initialization
Input Files Read (timestamp1)
Grid Configuration (timestamp2)
Domain Decomposition (timestamp3)
Variables memory alloc. (timestamp4)
Pre processing initial state calculation (timestamp5)
Plot and history files initialization (timestamp6)
Initialization time measure (timestamp7)
Atmosphere 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

results.

4.1 Experimental Results

From previous works, (Osthoff et. all.(2010) and (Schepke et. al. (2010), we know that the OLAM scalability on SunHPC system is limited by the performance of the output operations and that using many cores in the same node of a multicore system further penalizes the performance of these output operations. Therefore, there is an maximum number of cores that each SunHPC platform machine can use for a given OLAM configuration that maximizes the performance in face of a trade-off between the benefit due to parallel processing on a greater number of cores and the penalization for having to use more cores in the same node.

This work presents two experimental analyses evaluating the maximum number of cores per SunHPC node in two distinct OLAM workload configurations. In the first experiment we performed a 200km OLAM configuration typical analysis. The details of this configuration were presented in Section 3.3.

The second experiment performed a 40km OLAM configuration typical analysis. Therefore, we decreased 5 times the horizontal distance between points in the globe while the others parameters were the same from the 200km configuration. For this simulation to cover the same horizontal area as before (the globe), it is necessary 25 times the number of points, resulting in 25 times more calculations per timestep. Furthermore, it implies a 10-fold increase in the memory workload, thus increasing the dependency relationship between computing time and workload transfers.

4.1.1 200km OLAM Experimental Results

Figure 3 presents optimum and measured speed-up from 1 to 64 cores for the 200km OLAM configuration running 8 cores in each machine. In order to evaluate the maximum number of cores in this setting, we investigate the speed-up curve from 1 to 8 cores, presented in Figure 4. We observe that, for this configuration, the number of cores that keeps the curve closest to the

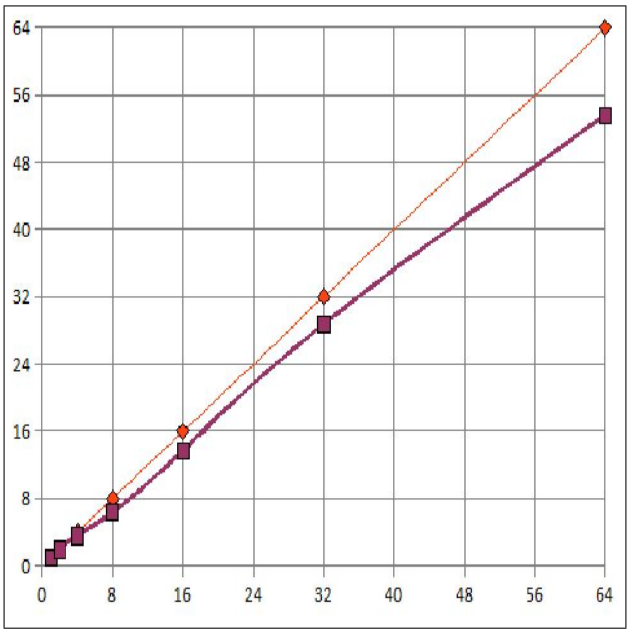

Figure 3: SunHPC Cluster Multicore 200km OLAM Speed-up

ideal curve (at a distance of 10%) is 4 cores.

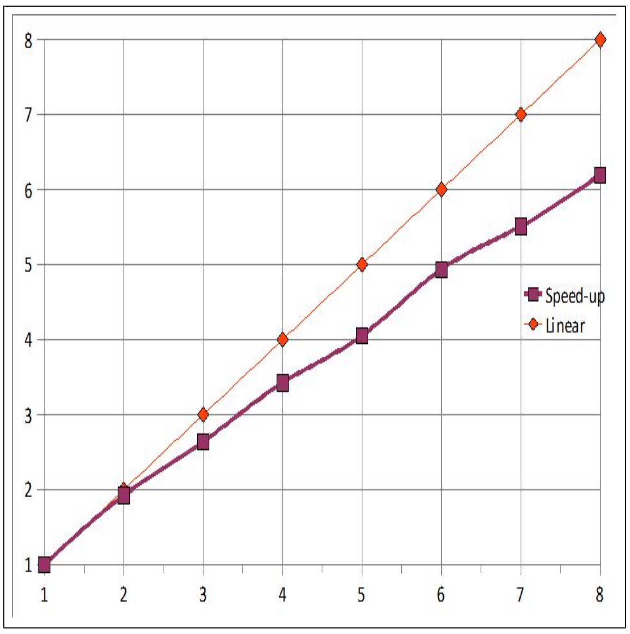

Figure 4: SunHPC 8 cores per machine 200km OLAM Speed-up

Figure 5 presents the 200km OLAM 4 core speed-up from 1 to 64 cores. By contrasting Figures 3 and 5, one observes that running OLAM with 4 cores per node increases the scalability of the system. Therefore, we observe that, for more than 64 cores, the scalability starts to decrease. This indicates that there are other problems degrading system output performance.

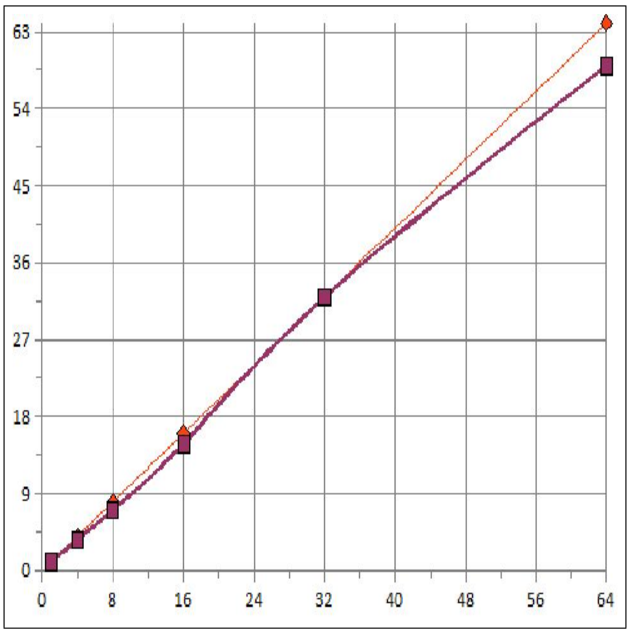

Figure 5: SunHPC 4 cores per machine 200km OLAM Speed-up

4.1.2 40km OLAM Experimental Results

In this section we search for the maximum number of cores per node for the 40km OLAM configuration. Following the same procedure as before, we analyze the speed-up curve from 1 to 8 cores, presented in Figure 6, we observe that, for this configuration, the maximum number of cores that improves the scalability of OLAM on this cluster platform is 2. Figure 7 presents 40km OLAM 2 core per machine speed-up from 1 to 64 cores on SunHPC platform. As we expect, running with 2 cores per machine increases the scalability of the system, also for more than 64 cores the scalability starts to decrease.

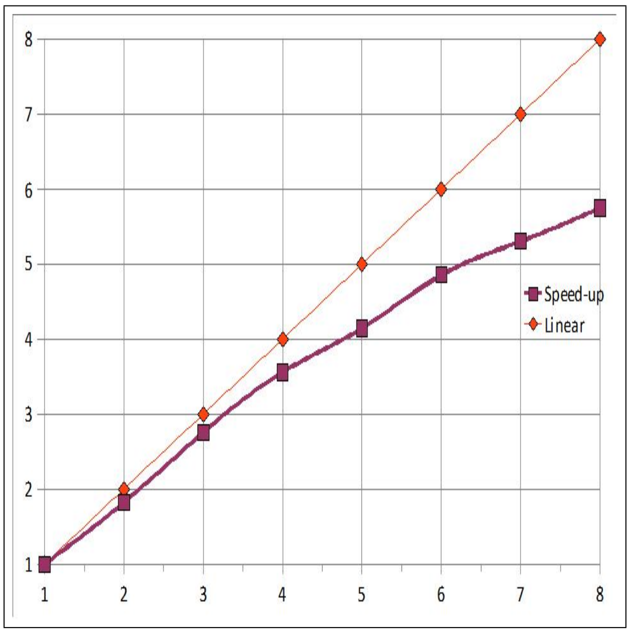

Figure 6: SunHPC 8 cores per machine 40km OLAM Speed-up

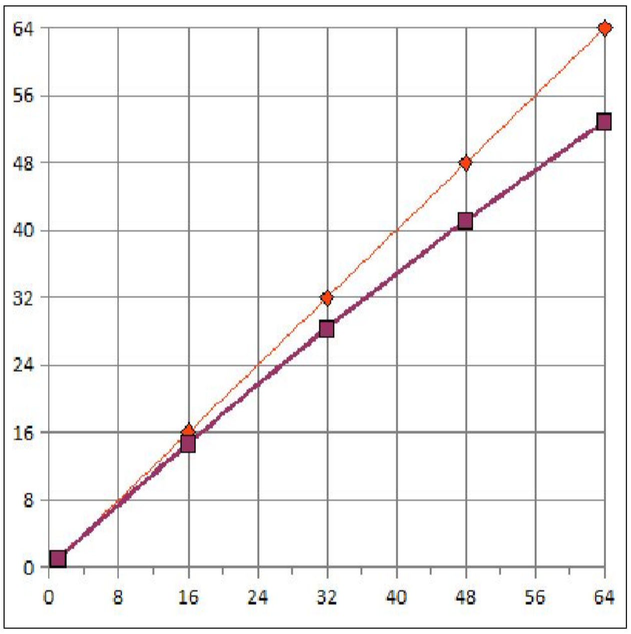

Figure 7: SunHPC 2 cores per machine 40km OLAM Speed-up

Table 1: OLAM configuration speed-up for 64 cores run.

Olam Configuration	Speed-up
200km - 8 cores per machine	53.62
200km - 4 cores per machine	59
200km - 2 cores per machine	59,6
40km - 8 cores per machine	40.64
40km - 4 cores per machine	49.85
40km - 2 cores per machine	52,89

4.2 Experimental Analysis

These two experiments show that for a given OLAM configuration there is an maximum number of cores in order to improve the speed-up. Also, we observe that for more than 64 cores the scalability for both experiments starts to decrease, indicating the existence of other problems degrading system output performance.

From previous work, we know that the overhead of I/O operations is aggravated by the OLAM workload pattern which accesses a large number of distinct files. This pattern generates simultaneous write operations in different files, overloading the I/O nodes system. We plan to investigate this problem in future work.

Table 1 presents the speed-up for 64 cores run in three machine configurations for the two OLAM settings. For a given number of cores per machine, we observe that the 200km OLAM configuration performance is better than the 40km OLAM configuration performance indicating that OLAM memory workload is the bottleneck on SunHPC platform multicore system. In order to explain OLAM multicore memory contention, table 2 presents Vtune Analyzer parameters for OLAM configuration running 8 cores in the same machine and running 8 cores in one different machine each.

Table 2 presents Vtune L2 cache miss and Bus Transfer Memory percentage parameters

Table 2: OLAM configuration Vtune percentage parameters

Olam Configuration	L2 cache miss	Bus Trans Memory
200km-8cores per machine	99,94 %	99,82 %
200km-1core per machine	70,16 %	71,325%

for the 200km OLAM configuration run with 8 cores per machine compared to 1 core per machine in the SunHPC. We observe that as we increase from 1 core per machine to 8 cores per machine the parameters L2 cache misses and Bus Transfers Memory increase, indicating memory contention for 8 cores run.

Further Vtune investigation shows that the routines that dominate L2 cache misses are related to timestep calculations. Therefore we need to investigate the timestep calculation algorithm for memory data allocation in order to decrease L2 cache misses.

5 CONCLUSIONS

This work evaluates the maximum number of cores according to two Ocean-Land-Atmosphere Model (OLAM) workload settings on a multicore cluster environment.

Our experiments show that for a given OLAM configuration there is an maximum number of cores in order to improve a given multicore cluster platform speed-up. Also, we observe that for more than 64 cores the scalability for both experiments starts to decrease, indicating the existence of other problems degrading the system output performance.

From previous work, we know that the overhead of I/O operations is aggravated by the OLAM workload pattern which accesses a large number of distinct files. This pattern generates simultaneous write operations in different files, overloading the I/O nodes system.

For future works we envision several directions. For instance, we plan further investigating in order to evaluate the relationship between maximum number of cores according to Atmospheric Model application workload parameters.

In order to decrease OLAM multicore memory contention we plan to further investigate the memory data allocation algorithms in the routines that dominate L2 cache misses.

We also intend to evaluate the parallel I/O library from (Marshall et. al. (2009) aiming to reduce OLAM initialization file creation overhead and the work of (Nawab et. al.(2009) aiming to improve output write operations performance. Besides, we plan to study the small file I/O techniques from (Carns et. al. (2009). Another work that, we expect, will help us improve OLAM I/O performance is (Ohta et. al. (2008). They present an approach to have many simultaneous I/O requests gathered, buffered locally, reordered to reduce disk seek times and then scattered to I/O nodes in parallel.

REFERENCES

Adcroft, A. ,Hill,C.,Marshall *Representation of topography by shaved cells in a height coordinate ocean model.* Man.Wea.Rev., pages 125,2293-2315,1997.

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

Datta, K., et all. *Stencil Computation Optimization and Auto-tunning on State-of-the-Art Multicore Architectures* Conference on High Performance Networking and Computing Procedings of the 2008 ACM/IEEE conference on Supercomputing, Section Papers, Article n 4, 2008

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.

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.

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.

Ohta, K., Matsuba, H. And Ishikawa, Y. *Gather-Arrange-Scatter: Node-Level Request Reordering for Parallel File Systems on Multi-Core Clusters* IEEE International Conference on Cluster Computing, pages 336-341 , 2008.

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

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), 2010.

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

Shalf, J. *Memory Subsystem Performance and QuadCore Predictions* Presentation at NERSC User Group Meeting September 17, 2007.

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

Wenneker,I.,Segal,A.,Wesseling P. *A Mach-uniform unstructured staggered grid method.* J . Numer. Meth. Fluids, 1209-1235, DOI:10.1002/fld., pages 417-202, 2002