Project Number GA: **228203**

Project Acronym: **IS-ENES**

Project Title: **Infrastructure for the European Network for Earth System Modelling**

Programme: **SEVENTH FRAMEWORK PROGRAMME**
Capacities Specific Programme
Research Infrastructures

D8.2 - Report on the Description of the Evaluation Suite and Base-case Results

WP8/JRA2: *European ESM: Performance Enhancement*

Due Date:		M18
Submission Date:		-
Start Date of Project:		01/03/2009
Duration of Project:		48 months
Organisation Responsible for the Deliverable:		BSC (16)
Version:		1
Status		Final
Author(s):	O. Jorba – M. Val – F. Martínez – D. Vicente	BSC (16)
	G. Riley	UNIMAN (6)
	C. Basu	LIU (15)
	A. Caubel	CNRS-IPSL (1)
	E. Maisonnave	CERFACS (3)
	I. Epicoco	University of Salento
	S. Mocavero	CMCC
	G. Aloisio	CMCC - University of Salento

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	16/06/2010	First draft: definition of the outline	O. Jorba, G. Riley
0.2	26/07/2010	Second draft: report compilation	C. Basu, A. Caubel, O. Jorba, E. Maisonnave, F. Martínez, M. Val, D. Vicente
0.3	01/08/2010	Third draft: report compilation, including the CMCC contribution	C. Basu, A. Caubel, O. Jorba, E. Maisonnave, F. Martínez, M. Val, D. Vicente, I. Epicoco, S. Mocavero, G. Aloisio
0.4	16/08/2010	Fourth draft: report with complete CMCC contribution and CERFACS comments	G. Aloisio, C. Basu, A. Caubel, I. Epicoco, O. Jorba, E. Maisonnave, F. Martínez, S. Mocavero, M. Val, D. Vicente
0.5	24/08/2010	Fifth draft: report including comments by Sophie Valcke	G. Aloisio, C. Basu, A. Caubel, I. Epicoco, O. Jorba, E. Maisonnave, F. Martínez, S. Mocavero, M. Val, S. Valcke, D. Vicente
1	14/09/2010	Final report, including addressed comments made by Giovanni Aloisio	G. Aloisio, C. Basu, A. Caubel, I. Epicoco, O. Jorba, E. Maisonnave, F. Martínez, S. Mocavero, M. Val, S. Valcke, D. Vicente, G. Riley

Table of Contents

PART I: Overview

1. Executive summary

The W8/JRA2 work package undertakes research into the performance aspects of configuring, deploying and running Earth System Models (ESMs). In deliverable WP8/JRA2 D8.1 “Definition of the Evaluation Suite”, a set of ESMs and stand-alone models available in the IS-ENES consortium were documented. This deliverable reports on work in which selected models from the evaluation suite have been ported and tested on a number of HPC infrastructures available to the IS-ENES partners, along with some additional activities. A collaborative effort between application owners and computer specialists has led to the identification of numerous current strengths and limitations of ESMs.

The IS-ENES evaluation suite consists of five coupled models: CMCC-MED, ARPEGE-NEMO, IPSL-ESM, HadGEM2 and EC-Earth, in addition to four stand-alone models: ARPEGE, NEMO, LMDZOR and ECHAM5. Detailed technical information on these models can be found in D8.1. Specific details of the models reported in this document are given in the associated technical reports in Part II.

The report is organized in three main parts: Part I (this section) gives an overview and introduces the report, presents its purpose and summarises the activities undertaken; Part II is a compendium of reports presenting the base-case results and consists of a set of technical reports on each activity undertaken, and Part III draws conclusions and discusses future work that aims to improve the performance of the models on current and future computing resources.

2. Introduction

The performance analysis of current Earth System Models (ESMs) on state-of-the-art computing systems is undertaken in work package WP8/JRA2. The increase in computational resources associated with the ongoing work done in DEISA, DEISA2 and PRACE projects stresses the need to improve the current performance of ESMs on HPC systems. Climate science has made major steps in modelling the evolution of the climate through complex coupled models. However, current coupled models appear not to be suited to exploit fully the HPC resources that are planned to be deployed in next few years. For example, Moore's law suggests that the first exascale computer will be available in 2018 and this is likely to consist of 'billions' of cores. In the IS-ENES project, Task 8.2, "Portability, performance analysis and improvement", seeks to understand and improve the performance of ESM models for current and future HPC systems. This task focuses on the performance aspects of both individual component models and ESMs constructed from them with the purpose of ensuring the ESMs can execute efficiently on existing large-scale computing facilities and also that the plans to prepare the models for execution on future facilities are developed. Particular attention is given to ensuring models will be able to take advantage of the PRACE initiative and DEISA2.

2.1 Purpose

The purpose of the present report is to summarize the work undertaken to understand the performance of the ESMs of the evaluation suite on current HPC infrastructures. The evaluation suite is defined in report "D8.1: Definition of the Evaluation Suite" of work package WP8/JRA2. Several activities have been undertaken to port, test and evaluate, using profiling and trace analysis, for example, ESM model performance on current state-of-the-art computing systems. The works aims to document:

- The current performance of the models on existing parallel architectures,
- critical aspects that climate models stress in current HPC architectures,
- bottlenecks, and strengths and weaknesses of each of the models in order to guide the design and development of future optimized ESMs for the upcoming peta- and exascale architectures.

2.2 Glossary of Acronyms

Acronym	Definition
ESM	Earth System Model
DEISA	Distributed European Infrastructure for Supercomputing Applications
PRACE	Partnership for Advanced Computing in Europe
HPC	High Performance Computing

Table 1: Glossary of Acronyms

3. List of activities

The activities undertaken in JRA2 during the first part of the project IS-ENES have had as main objective to evaluate the current status of performance of several Earth System Models. The following list summarises the reports describing the work done to characterize such performance in current HPC infrastructures:

- LIU: NEMO Porting, Benchmarking and Tuning on Linux Cluster. C. Basu
- IPSL, BSC, CERFACS: IPSL-ESM (LMDZOR-NEMO-OASIS3) porting and performance tests. A.Caubel, O.Jorba, E. Maisonnave
- CERFACS: Development of a high resolution version of ARPEGE-NEMO climate model. E. Maisonnave
- CERFACS, IPSL, BSC: ARPEGE-NEMO porting and performance tests on DEISA-PRACE-GENCI platforms. E.Maisonnave, A.Caubel, O.Jorba
- BSC: Porting and performance analysis of the EC-EARTH model on MareNostrum Supercomputer. F. Martínez, D.Vicente, O.Jorba, M.Val
- CMCC: Performance analysis and porting of the CMCC-MED model. I.Epicoco, S.Mocavero, G.Aloisio

The second part of the present document compiles the documents that report the work done and the main conclusions of each initiative. Results of the work will help to define the future work to undertake to improve the model performance in future HPC environments.

PART II: Base-case results

1. NEMO Porting, benchmarking and tuning on Linux Cluster

Author: Chandan Basu[1]

[1] National Supercomputer Centre, Linköping University, Sweden (LIU)

1.1 Introduction

At NSC we are working on porting, optimization and tuning of NEMO program on linux - X86-64 - infiniband clusters. As many of the modern day clusters fall in this category our benchmarking will be indicative of NEMO performance on these types of clusters. The performance of any MPI program may depend on various components, e.g., compilers, MPI libraries, network, filesystem, processor technologies etc. As NEMO is a large and complex program its performance needs to be tested against these variables. In the first phase of our optimization work we are looking into these factors for an optimized run. We are also carrying out scaling test and profiling of the codes. The goal is to figure out the bottlenecks and to look at suitable strategies to improve the timing of the code. We have used ORCA1 configuration available from NOCS website (http://www.noc.soton.ac.uk/nemo/) for our scaling / benchmark studies. The ORCA1 configuration is bigger than default ORCA2 configuration that comes with NEMO. So it is more suitable for benchmarking studies on large number of nodes. Therefore, this benchmark will give us a fair idea about scaling behaviour of NEMO. Also it seemed to us that ORCA1 resolution is reasonable for many practical situations.

Test systems: We use two systems for our benchmarking. The details of these systems are given in Table 1.1.

System	Ekman	Kappa
Processor	Quad-Core AMD Opteron 2374 HE @ 2.19 GHz	Quad core Intel(R) Xeon(R) CPU E5520 @ 2.27GHz, code named Nehalem
Interconnect	DDR Infiniband, full bisection bandwidth	DDR infiniband, reduced bisection bandwidth
Node	2 processor, 8 core	2 processor, 8 core
No. of nodes	1268	364
MPI processes / node	8	8
global filesystem	Lustre over infiniband	GPFS over gigabit ethernet
Compiler	ifort - 11.0.074, pgf90 - 10.3	ifort - 11.1.059
MPI	Open MPI - 1.4.1, Scali MPI - 5.6.6	Open MPI - 1.4.1

Table 1.1: Description of systems used for benchmarking

1.2 Scaling of NEMO for ORCA1 benchmark

We have run NEMO for 1 year period on different number of processors to see its scaling behaviour for ORCA1 configuration. The results are given in Table 1.2 and Figures 1.1 and 1.2. We see that ORCA1 run scales up to 16 nodes (128 cores) on both Ekman and Kappa systems. However on Kappa the runs are faster than on Ekman. This is owing to the fact that Kappa has latest Intel Nehalem processors compared to Ekman which has AMD Opteron processor. The memory performance of Nehalem system is better than Opteron.

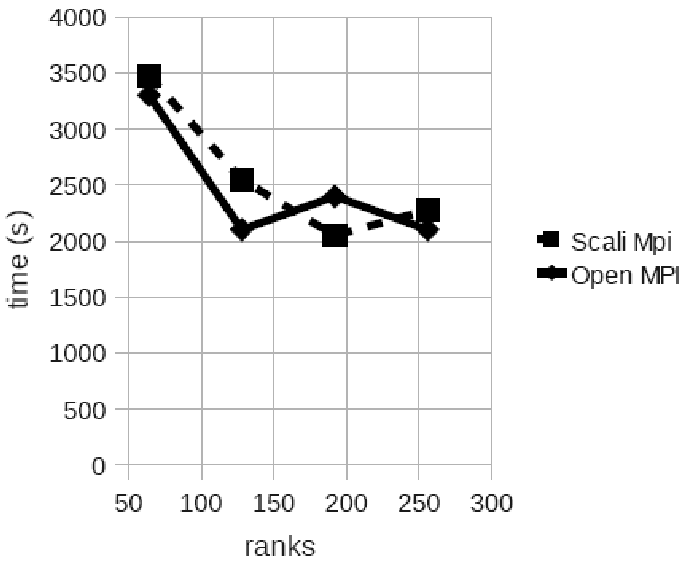


Figure 1.1: ORCA1 runs on Ekman

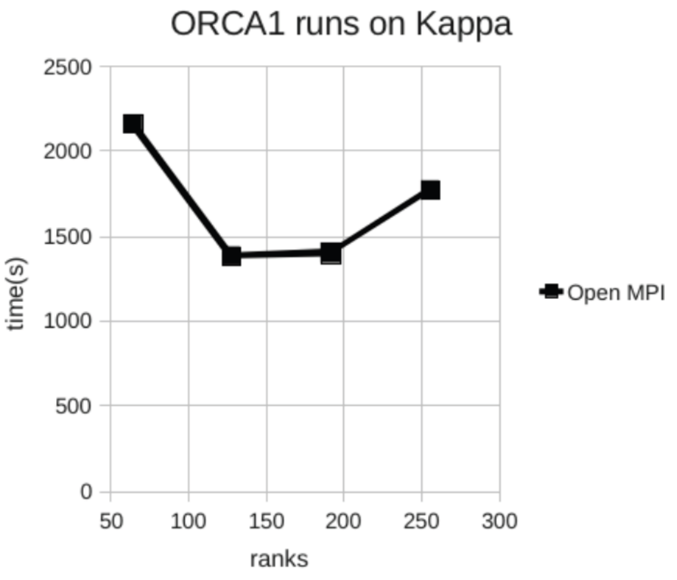


Figure 1.2: ORCA1 runs on Kappa

System	Grid	No. of ranks	MPI	time (s)
Ekman	8X8	64	Scali MPI	3478
Ekman	8X16	128	Scali MPI	2548
Ekman	12X16	192	Scali MPI	2049
Ekman	16X16	256	Scali MPI	2274
Ekman	8X8	64	Open MPI	3301
Ekman	8X16	128	Open MPI	2101
Ekman	12X16	192	Open MPI	2397
Ekman	16X16	256	Open MPI	1674
Kappa	8X8	64	Open MPI	2162
Kappa	8X16	128	Open MPI	1384
Kappa	12X16	192	Open MPI	1403
Kappa	16X16	256	Open MPI	1771

Table 1.2 Scaling studies of ORCA1 benchmark

1.3 Tuning of NEMO runs

We have looked into various factors for optimized NEMO runs. These are described below.

Effect of different compilers: We have compiled NEMO 3.2 with Intel compiler and PGI compiler on Ekman system (see Table 1.1 for details of system and compilers). We generally observed that Intel compiled binaries work faster than PGI compiled binaries on Ekman system. On Kappa system we do not have PGI compilers installed. So we have done all our runs using Intel compiled binaries.

Effect of process binding: MPI process running on a node can be swapped between available cores by Linux OS scheduler. This generally degrades performance. It is however possible to force each MPI processes to run on specified core. This is termed as process binding. Process binding can result in optimized cache performance resulting in better performance. We have tested processor binding effects by binding the processes and then comparing timing for runs without binding. We see that in general processor binding gives 1 - 2 % better run-time (See Table 1.3) for almost all runs. The processor binding can be achieved through mpirun by passing suitable flags to it. However we have observed that for different MPIs the processor binding through mpirun does not work always. We have also seen that in some cases mpirun binds processes wrongly (e.g., more than two processes bound to same core!). So we do not use mpirun to bind processes. Instead we fire jobs normally without binding and then use Linux command taskset to explicitly bind each process to a distinct core.

System	Grid	no. of ranks	MPI	Time (s)	
				Proc. binding disabled	Proc. binding enabled
Ekman	8X8	64	Scali MPI	3478	3391
Ekman	8X16	128	Scali MPI	2448	2357
Ekman	12X16	192	Scali MPI	2049	2001
Ekman	16X16	256	Scali MPI	2274	2265
Ekman	8X8	64	Open MPI	3301	3231
Ekman	8X16	128	Open MPI	2101	2069
Ekman	12X16	192	Open MPI	2397	2304
Ekman	16X16	256	Open MPI	1674	1642
Kappa	8X8	64	Open MPI	2162	2125
Kappa	8X16	128	Open MPI	1384	1360
Kappa	12X16	192	Open MPI	1403	1400
Kappa	16X16	256	Open MPI	1771	1760

Table 1.3 Scaling studies of ORCA1 benchmark with processor binding

Effects of changing MPI libraries: Scaling of parallel programs depends on MPI communication routines. So changing from one MPI implementation to another might have some impact on the overall performance of NEMO. We have compared NEMO runs with Open MPI & Scali MPI (see Table 1.4). We see that there is 5 - 15 % change in runtime of NEMO by changing MPI library. We also see that for 8, 16 and 32 node jobs Open MPI runs are faster but for 24 node job Scali MPI runs are faster.

t_scali = Scali MPI run time

t_ompi = Open MPI run time

dt = (t_scali - t_ompi) / t_ompi

System	**Grid**	**no. of ranks**	**% dt**
Ekman	8X8	64	5.36
Ekman	8X16	128	14.67
Ekman	12X16	192	-14.56
Ekman	16X16	256	35.05

Table 1.4 Effect of different MPI libraries on ORCA1 runtime

Effects of filesystem: NEMO ORCA1 configuration does frequent of i/o while running. Normally MPI jobs are fired from global file systems. The files in these filesystems are visible from all the compute nodes. As all the nodes try to read / write data via some network this can be a performance bottleneck for jobs which does lots of i/o. Also in a large cluster where many jobs are running file i/o throughput of global filesystems can be impacted by the overall i/o and network congestion. We have done some testing with firing jobs from local disks. In this case we copy all the input data to the local hard disk of every node. The jobs are fired from local disk. While running the jobs do i/o to local disks. Thus the i/o becomes parallel. Also i/o speed is not dependent on the network. At the end of the job all the output files written on local disks are copied back to some global filesystem. In this case the significant time is mpirun time + data copy time. Our general observation is that for Ekman system the jobs fired from local filesystem takes almost the same time as the jobs fired from global filesystem. But on Kappa system the local runs are considerably faster than the runs from global filesystem (see Table 1.5).

del t = (tg - tl) / tl

tg = run time from global

tl = run time from local + data scatter, gather time

System	Grid	No. of ranks	MPI	% del t
Ekman	8X8	64	Scali MPI	1.9
Ekman	8X16	128	Scali MPI	1.5
Ekman	12X16	192	Scali MPI	2.0
Ekman	16X16	256	Scali MPI	2.3
Kappa	8X8	64	Open MPI	23.4
Kappa	8X16	128	Open MPI	25.1
Kappa	12X16	192	Open MPI	20.1
Kappa	16X16	256	Open MPI	10.5

Table 1.5 Impact of using local vs global file system on ORCA1 runtime

The difference in behaviour between Ekman and Kappa system could be due to relative speed of local and global filesystem. We can see that running from local filesystem in general gives better timing. Also sometimes global filesystem can be slow depending on the load on the file system. On the other hand it is bit complicated to move data back forth to and from local filesystem. But this can be achieved through some script which will copy the data to node local file system and after the job is over will copy back relevant output files. If the system has parallel shell commands, e.g., pdcp, pdsh etc. available then the file copying can be done quite efficiently.

Static linking vs dynamic linking: Static linking of MPI libraries generally leads to improvement in the run time. This is shown in the Table 1.6

t_dyn = run time for dynamic linking

t_stat = run time for static linking

del t = (t_dyn – t_stat) / t_stat

System	Grid	No. of ranks	MPI	% del t
Ekman	8X8	64	Open MPI	10.5
Ekman	8X16	128	Open MPI	6.3
Ekman	16X16	256	Open MPI	3.1

Table 1.6 Impact of using static vs dynamic linking on ORCA1 runtime

1.4 Profiling of NEMO

As we have seen above NEMO run with the ORCA1 benchmark does not scale well beyond 16 nodes (128 cores). We believe that profiling of the code will give us more insights about how scaling is lost when we increase the no of ranks. We have profiled the MPI calls in NEMO using Scali MPI's profiling feature. We start the profiling after initialization is over. We profile the code when it is in the main time loop. The Table 1.7 below shows average time spent in significant MPI calls for several 2 minute window of run-time. As we can see for 64 rank run time spent in MPI calls is 44 %, for 128 rank run the time spent in MPI calls is 66 % and for 256 rank run 82 % of time is spent in MPI calls within the main time loop of NEMO run. This is because the number of MPI calls for each rank increases for wider NEMO runs. Consequently the delays associated with data transfer increases. Another factor which may lead to increase in MPI times is load imbalance between different ranks.

MPI call	64 rank			128 rank			256 rank		
	No. of calls	**Time (s)**	**delta (%)**	**No. of calls**	**Time (s)**	**delta (%)**	**No. of calls**	**Time (s)**	**delta (%)**
MPI_Allreduce	97970	7.2	~10	162030	17.1	~12	173724	19.5	~15
MPI_Isend	517320	0.9		853223	1.5		921234	1.3	
MPI_Recv	517320	36.0		853223	47.5		921234	61.4	
MPI_Wait	517320	0.05		853223	0.1		921234	0.1	
Sum	1649930	44.0		2721700	66.2		2937426	82.3	

Table 1.7 Profiling of MPI calls in Nemo

The delta parameter in the Table 1.7 refers to the deviation in MPI time between ranks. This difference in MPI times could be due to load imbalance in different ranks. As we can see that the delta value increases with increasing number of ranks. This may also lead to increase the waiting time for MPI calls to finish.

1.5 Conclusions

We have run NEMO on Ekman and Kappa system under various conditions. We see that NEMO compiles with both Intel as well as PGI Fortran. It can be linked with Open MPI or Scali MPI. We have observed that on Ekman system Intel compiled NEMO works faster than PGI compiled NEMO.

The ORCA1 benchmark scales to around 16 nodes (128 cores) on both Ekman as well as Kappa system. Although on Ekman there is some scaling up to 32 nodes (256 cores). The better scaling in Ekman may be due to the fact that single node performance of Ekman is slower than Kappa. Normally slower nodes scale better than faster nodes if the interconnect speed remains same. For higher number of node count large amount of the time is spent in MPI calls. So the scaling is gradually lost. For a bigger configuration NEMO will probably scale to more number of nodes.

For an optimized run it is better to bind MPI processes to cores as we have seen in our results (Table 1.3). But the improvements are marginal and default scheduler of Linux operating system seems to work quiet efficiently. Also processor binding should be done carefully to avoid wrong processor binding.

Choice of MPI library seems to have some impact on run time of NEMO. In our runs Open MPI runs are consistently better than Scali MPI runs (Table 1.4) except in the case when number of cores are 192. The behaviour may be due to change of some MPI internal algorithm at certain sizes. Also for ORCA1 configuration the good scaling is up to 128 cores. Beyond 128 cores the code is running inefficiently and the difference in runtime may not be indicative of just MPI performance.

Statically binding MPI library gives better performance than dynamic binding (Table 1.6). This has been verified for Open MPI.

Firing NEMO from local filesystem gives better timing than firing from global filesystem (Table 1.5). However, the extent of difference in timing seems to depend on the relative speed of local vs global filesystem. As we can see that in Kappa system there is substantial improvement in runtime but in Ekman system there is only marginal time improvement.

We have described above some strategies for optimized NEMO runs which should be applicable for similar systems. There may be further scope of improving the timing and scaling. But for this we have to look inside the code.

We have already done some profiling of MPI calls in NEMO and we would like to continue profiling of NEMO including MPI calls as well as other functions and subroutines. We will use different available tools for profiling. We also plan to put some light weight timers in the code to analyze its behaviour. The correct profiling will tell us accurately about load imbalance if any and other bottlenecks..

2. IPSL (LMDZOR-NEMO-OASIS3) Earth System Model and LMDZOR and ARPEGE models porting and performance tests

Authors: A. Caubel[1], O. Jorba[2], E. Maisonnave[3]

[1] Institut Pierre Simon Laplace, France (IPSL)

[2] Barcelona Supercomputing Center, Spain (BSC)

[3] Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, France (CERFACS)

2.1 Introduction

The present IPSL (Institut Pierre Simon Laplace) model couples four components of the Earth System: LMDZ for atmospheric dynamics and physics, ORCA for ocean dynamics, LIM for sea-ice dynamics and thermodynamics, and ORCHIDEE as the land surface component. It can include a full carbon cycle component, as well as a model of the atmospheric chemistry and aerosols. The OASIS coupler is used to synchronize, interpolate and exchange fields between atmospheric and oceanic components.

Actual IPSL scientific production is done on vector machines (with weak parallelism). That's why studying the behavior of IPSL models at higher resolution than the current one on DEISA-PRACE-GENCI-like machines (SMP and MPP scalar machines) could be a good indicator of both work done recently at IPSL on parallelization aspects and porting to massively parallel supercomputers and current performance of IPSL models.

The set up of high resolution configuration of IPSL climate model started at BSC. A test configuration has run and allowed us to evaluate the performance but also the problems encountered when the resolution of each component is increased and the components coupled together. The configuration was atmosphere-land surface LMDZOR 280x280x19 on 72 CPUs (MPI parallelization) coupled to ocean-sea ice NEMO ½ degree (511x722x31) on 20 cores (MPI parallelization). This was a preliminary experimentation. We are now working at GENCI-CINES centre, on setting up a configuration at higher resolution: ¼ degree for oceanic model (1442x1021x75) and 1/3 degree for atmospheric model (768x768x39) using MPI-OPenMP parallelization.

This report describes the encountered difficulties during the model porting phase on MareNostrum IBM JS21 (BSC) and SGI Altix ICE (CINES) and some details scalar/vector compared performances.

2.2 BSC MareNostrum IBM JS21

2.2.1 Machine description

The Barcelona Supercomputing Center (BSC) hosts MareNostrum, one of the most powerful supercomputer in Europe and the number 87 in the world [Top500 list, June 2010]. MareNostrum was built in March 2004 as a result of an agreement between the Spanish government and IBM. MareNostrum is a node of DEISA2 consortium and BSC is currently a partner in the PRACE project. In this sense, MareNostrum is targeted as one of the first DEISA2 HPC facilities to test ESMs.

The MareNostrum supercomputer is based on processors PowerPC, architecture BladeCenter, Linux operating system and Myrinet interconnection. MareNostrum has 10240

IBM Power PC 970MP processors, 20 TB of main memory and 280 + 90 TB of disk storage. It uses two interconnection networks: Myrinet and Gigabit Ethernet. It is the first supercomputer that runs under a Linux operating system, a SuSe Distribution. The Peak Performance of the system is 94.21 Teraflops.

Marenostrum has 44 racks, 31 of them dedicated to computing tasks. The computing racks have a total of 10240 processors. Each rack is formed by 6 Blade Centers. In total, each rack has 336 processors and 672 Gb of memory; each one has a rough peak performance of 3.1 Tflops.

Each Blade Center has 14 server blades type JS21. Each of these nodes has 2 processors PowerPC 970MP at 2.3 GHz, 8 Gb of shared memory between both processors and a local SAS disk of 36 Gb. Each node has a network card Myrinet type M3S-PCIXD-2-I for its connection to the high speed interconnection and the two connections to the network Gigabit. Each node has a local disk of 36 Gb and works diskless, i.e., the operating system is in the storage racks instead of the local disk and it is loaded through a Gigabit network when each node is initialized.

In addition to the local disk of each node, MareNostrum has 20 storage servers arranged in 7 racks. These servers have a total of 560 disks of 512GB and each one provides a total capacity of 280TB external storage. These disks are working with Global Parallel File System (GPFS), which offers a global vision of the file system and also allows a parallel access.

Default compilers in Marenostrum are IBM XL C/C++, and IBM XL FORTRAN. In addition, the GNU C and FORTRAN compilers are available.

Several numerical libraries and several application packages are installed in MareNostrum (http://www.bsc.es/plantillaC.php?cat_id=472).

2.2.2 Achievements for the whole IPSL coupled system

To better understand the behaviour of IPSL models on a machine MareNostrum-like, several tests have been done. Two significant configurations were created and tested on this machine and thanks to the timer implemented in Oasis3 by Eric Maisonnave, CERFACS (see Chapter 4: *ARPEGE-NEMO porting and performance tests on DEISA-PRACE-GENCI platforms*), we obtained the following performances:

A high resolution coupled configuration:

This version has been set up to evaluate the performance but also the problems encountered when the resolution of each component is increased and the component coupled together. This configuration is based on the following components and resolutions:

- atmosphere-land surface LMDZOR 280x280x19 on 72 CPUs
- ocean-sea ice NEMO ORCA05 (511x722x31) on 20 CPUs

Using a “pseudo-parallel” version of the OASIS3 coupler on 2 processors, we obtained the following performances: 5-day simulation in 21 min 10s (i.e., 10-year simulation in 10 days).

More details and results:

Real time for LMDZOR configuration (forced mode) to run a coupling period (i.e., 1 day) is 220s by using 72CPUs.

Real time for NEMO component (1-day simulation) at ORCA05 resolution:

Number of CPUs	5	10	20	27	32
Real Time NEMO	760s	382s	201s	141s	125s

Table 2.1: Real time of NEMO (1- day simulation) at ORCA05 resolution on MareNostrum

<u>An Earth system model configuration</u>:

This version has been set up to evaluate, by running a long simulation of an Earth system model on this type of architecture, the feasibility of a realistic climate simulation. This configuration is based on the following components and resolutions:

- atmosphere-land surface-atmospheric chemistry LMDZOR-INCA 96x71x19 on 24 CPUs

- ocean-seaice NEMO ORCA05 (511x722x31) on 50 CPUs

Using a “pseudo-parallel” version of the OASIS3 coupler on 2 processors, we obtained the following performances: 4-month simulation in 3h 22min. (i.e., 10-year simulation in 4 days)

More details and results:

Real time for NEMO (in forced mode) to run a coupling period (i.e., 1 day) is 95s by using 50 CPUs.

Real time for LMDZOR-INCA component (1-day simulation) at 96x71x19 resolution:

Nº of CPUs	10	14	16	24
Real Time LMDZOR-INCA	235s	197s	146s	116s

Table 2.2: Real time for LMDZOR-INCA component (1-day simulation) at 96x71x19 resolution on MareNostrum

2.2.3 Problems encountered

1) <u>Software environment</u>: Default versions of compiler and MPI libraries installed on MareNostrum do not allow running our current model versions. So, it is needed to use more recent compiler version and MPICH2 as MPI version.

2) <u>Heterogeneous codes</u>: we would like to run hybrid parallelization MPI-OpenMP for atmospheric component only and so, to have different specification (concerning parallel environment) for each executable, as follows:

    • Ocean (MPI parallelization)

    • Atmosphere (MPI-OpenMP parallelization)

    But, specifications are global on MareNostrum and it is not possible to specify requests depending on each component.

3) <u>Memory size</u>: Memory size problems occur when the resolution is increased, 280x280x19, for atmospheric component. An 8Gb (memory available per node) is not sufficient for more than 4 MPI processors. It is a problem both because the current version of LMDZ is not scalable in terms of memory and the LMDZ MPI-

parallelisation is limited by 3 bands of latitudes per MPI process.

LMDZOR resolution	96x72	144x143	192x192	280x280
Memory needed for 4 MPI process	2Gb	3.5Gb	6.7Gb	> 8Gb

Table 2.3: Memory size used by 4 LMDZOR MPI process on MareNostrum

2.2.4 Performances of atmospheric components LMDZOR and ARPEGE

Here is a comparison of performances on different kind of supercomputers. Since atmospheric component is the component which leads performances of the whole coupled model, we focused our analysis (i.e scalability, speedup) on the atmospheric model. Scalability of the whole coupled model is similar to the scalability of the forced atmospheric configuration.

The configuration used here is LMDZOR forced configuration (MPI parallelization) on the following machines:

- MareNostrum PowerPC (2,3 Ghz, 4 CPus/node, 8Gb/node)
- Vargas Power6 (4,7 Ghz, 32 CPus/node, 128-256Gb/node)
- Mercure SX8R (vector processors, 8 CPUs/node, 64Gb/node)

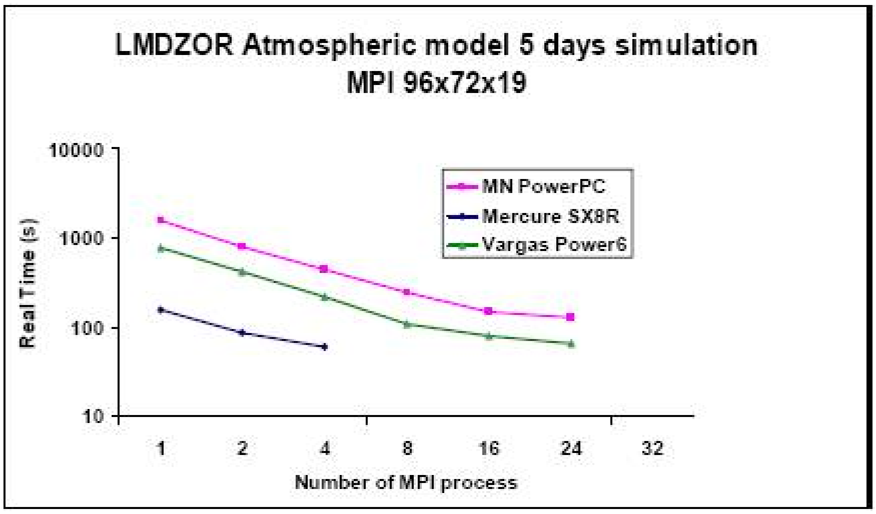


Figure 2.1: Comparison of Real Time inter-machine for LMDZOR at 96x72x19 resolution.

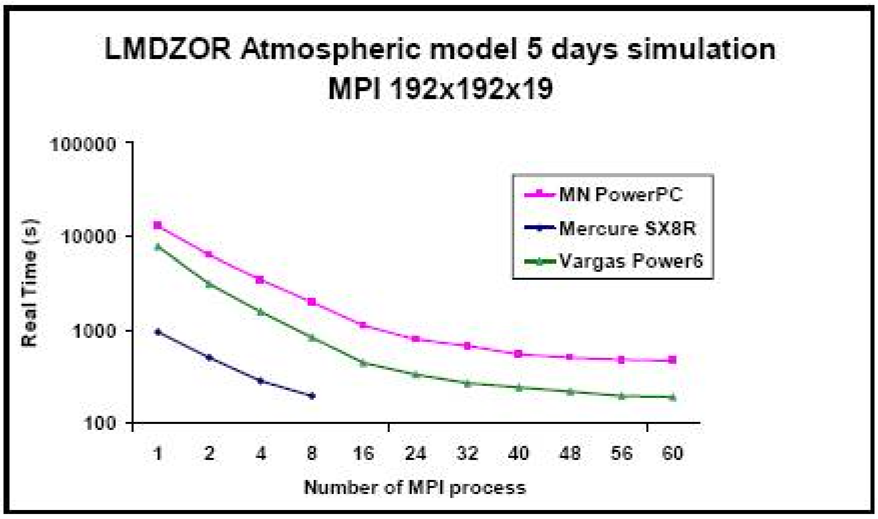


Figure 2.2: Comparison of Real Time inter-machine for LMDZOR at 192x192x19 resolution

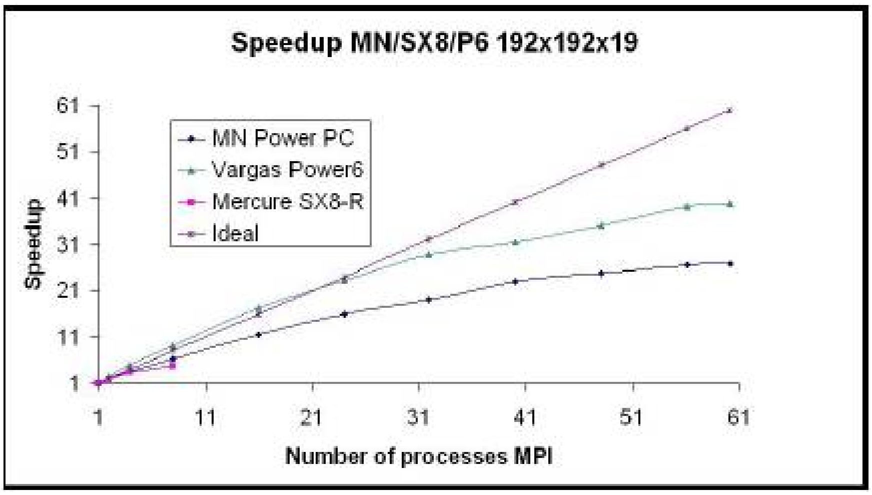


Figure 2.3: Comparison of speedup inter-machine for LMDZOR at 192x192x19 resolution

We can see, with the same number of MPI processes, that the real time is twice greater on the MareNostrum rather than on the Power6 Vargas. We can also note that the speedup is lower for PowerPC than for Power6. Different reasons can explain such differences regarding the performances: different architectures, frequency calculation of each processor (2,3Ghz for PowerPC processors and 4,7Ghz for Power6 processors), number of CPUs per node greater on Power6 (32 CPUs/node) than PowerPC (4 CPUs/node), etc.

Performances of LMDZOR configuration in MPI parallelisation are limited on PowerPC-like architecture in comparison with a Power6-like architecture. Even if we increase the number of CPUs used, real time is still greater in comparison with Power6.

We can obtain similar real time by increasing the number of CPUs on the Power6 in comparison with 4 or 8 vector processors SX8R. However, we never manage to obtain similar real time between PowerPC processors and Power6 processors (and of course SX8R processors).

More results and comparisons:

On mercure SX9, we run 10-year simulation in 1 day for the coupled configuration 96x95x39-ORCA2 on 4 CPUs.

On MareNostrum PPC, we run 10-year simulation in 10 days for the coupled configuration 280x280x19-ORCA05 on 94 CPUs.

	LMDZ **1 task/node**	**LMDZ** **4 tasks/node**	**ARPEGE** **4 tasks/node**
Instructions per cycle	0.62	0.47	0.33
Parallel efficiency	0.78	0.87	0.92

Table 2.4: Common LMDZ-ARPEGE analysis performances on MareNostrum (BSC)

We applied the PARAVER (http://www.bsc.es/plantillaA.php?cat_id=485) performance and analysis tool to compute the number of instructions per cycle and the parallel efficiency. If ARPEGE scalability exceed LMDZ' one, ARPEGE exhibits a slower number of instructions per cycle, which could imply a higher number of memory access conflicts. Parallel efficiency is slightly the same. All those figures could suggest that ARPEGE's higher scalability should be mainly originated in its smaller amount of communications (see Chapter 4: *ARPEGE-NEMO porting and performance tests on DEISA-PRACE-GENCI platforms*).

2.3 CINES ALTIX ICE

We are now working at GENCI-CINES centre, through the “Grand challenge” project, on setting up a configuration at higher resolution: ¼ degree for oceanic model (1442x1021x75) and 1/3 degree for atmospheric model (768x768x39) using MPI-OPenMP parallelization.

2.3.1 Description of the machine

The cluster SGI Altix ICE 8200, JADE, is a scalar parallel supercomputer with a peak performance of 147 Tflop/s. JADE is composed of 25 racks. One of these racks is used to ensure the connection with the whole cluster (via 3 login nodes) and 24 computing racks (cf. architecture’s description). This cluster consists of 12288 cores distributed on 1536 nodes (each node including 2 Intel Quad-Core E5472 processors). 30 GB of useful memory can be actually used on each node, i.e. over a total of 46 TB. The computing racks have access to a parallel file system Lustre with a total capacity of (509 TB).

2.3.2 Achievements

The work is still in progress but a first configuration has already run. The first results are as follows:

- Atmospheric model 0.3°x0.3°(39 vertical levels), 768x768x39 on 2048 cores by using hybrid MPI-OpenMP parallelization : 256 MPI processes x 8 OpenMP threads
- Oceanic model 0.25°x0.25°(75 vertical levels), 1441x1021x75 on 120 MPI processes
- Oasis3 parallel version “field per field” on 24 MPI processes (each MPI process treats one field).

⇒ 12 days are needed to run 10-year simulation on 2200 cores.

A 20-year simulation is planned to be done during next months to start to use such a configuration for scientific studies.

2.4 Conclusions

The porting of the IPSL climate model to the MareNostrum was very instructive both in terms of usability and performances. Firstly, it helped highlight the work done in recent years around the optimization and parallelization of IPSL models. Indeed, it is now possible to run a fully parallelized IPSL coupled model, which is essential on this type of architecture. Then, thanks to performance and analysis tools like PARAVER, we could see that this kind of architecture imposes some limits to our models and their performances: low memory available per node, scalar processors with low frequency calculation, moderate MPI scalability. Besides, the running environment seems to not be very suitable for heterogeneous codes: this would be a very strong constraint, especially since the parallelization hybrid MPI-OpenMP seems to be the best way to use up the resources available on MareNostrum-like architecture.

The second step done was the porting on CINES-Jade machine: the high resolution configuration tested on this machine was the final point of the work started at BSC-Marenostrum. The hybrid parallelisation MPI-OpenMP allows us to run our coupled model on around 2200 cores on this machine.

Acknowledgments:

Authors are grateful to support teams of computing centres (BSC and CINES). We would like to thank especially José M Baldasano and Oriol Jorba for the welcome and also IPSL Global Climate Modelling Group for the help and the common work.

3. Development of a high resolution version of ARPEGE-NEMO climate model

Author: E. Maisonnave[1]

[1] Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, France (CERFACS)

3.1 Model and machine description

Taking advantage of 2009 Météo-France NEC SX9 Operation Health Check (OHC), several configurations of ARPEGE-NEMO climate model have been set up in order to check OASIS3 capability at high end resolutions.

Tested resolutions were t359 (approximately 50Km square) with 31 vertical levels for ARPEGE and ORCA ¼ degree with 50 vertical levels for NEMO. Those resolutions are similar to the higher operational configurations in use at the moment in Europe (ECMWF, MetOffice, etc).

OASIS version 3, tag prism_2-3, was using for this experiment, and particularly the Arnaud Caubel's developments allowing parallel exchanges of coupling fields.

The new Météo-France supercomputer is composed by 6+7 vector nodes of 16 SX9 processors each of 102 Gflops peak performance and 1 Tb memory per node. Due to OHC time constraints, no fine optimization has been implemented. In particular, during simulation, data processing has not been done on local processor disks but through GPFS global file system.

3.2 Experimental design

Our coupling method implies that ocean and atmosphere perform simultaneously a coupled time step (namcouple LAG mode), each model using the coupled field averaged at the previous coupling time step.

Moreover, we used the MPI communication library with bufferized messages (namcouple NOBSEND option disabled), to let models continuing their calculations after sending their coupling fields to OASIS.

ARPEGE and NEMO are parallel codes: allocating appropriate processors number to each model, response time difference between each model could be minimized. However, due to vector processor efficiency (a fast simulation could be processed with less than 10 processors), possible combinations in processors repartition are limited and a model still remains significantly slower than the other. If OASIS duration (communication and interpolation processing) stays lower than this difference, there is no coupling additional time compared to slower stand alone simulation duration: OASIS calculations and communications are completed during the time interval between the end of fastest model and the end of slowest model calculations.

In order to measure those quantities, clock times are collected (thanks to light modifications under CPP key within OASIS “psmile” library) before and after each OASIS exchanges.

Measures are done on elapsed time and not on CPU time (OASIS CPU time are supposed to be independent of model parallelism ratio). The balance we proposed to tune here only influences elapsed time performances. But measures on elapsed time are machine load dependent, particularly if we share node with other users: an ensemble of 5 to 9 simulations (of 4 days of climate) will be processed and uncertainty evaluated.

Two kinds of measures are done to evaluate (a) the total coupled simulation time and (b) the calculation duration of each component of the coupled system.

A): Within OASIS, the difference between the very first “prism_get” of the whole simulation and the very last “prism_put” represents the whole climate simulation duration, excluding model restart read/write operations.

B): Within models, we measure the interval (at each coupling time step) between the instant after the “prism_get” of last coupling field received at coupling time step N and the instant before the “prism_get” of the first coupling field exchanged at time step N+1: in this way, we evaluate duration of calculation processed by each model between two calls to OASIS (supposing that the time for the non-blocking “prism_put” calls is negligible in this measure).

3.3 Optimizations and results

This counter allows us to finely determine the respective number of processors which minimizes the difference between oceanic and atmospheric coupling step durations. If 4 processors are allocated to NEMO (due to speedup optimum considerations), 6 to 7 processors for ARPEGE are most suitable. An important extra cost due to OASIS (more than 50%) incites us to optimize OASIS coupling technique.

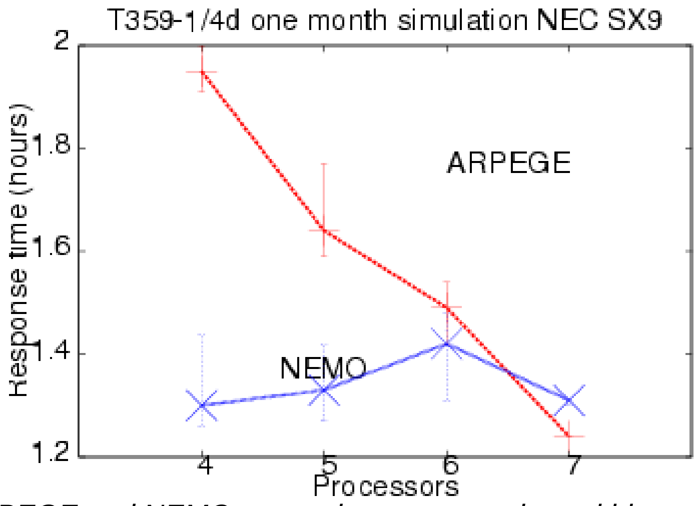


Figure 3.1: ARPEGE and NEMO respective response time within an ARPEGE-NEMO coupled model configuration (constant resources for NEMO -4- and varying number for ARPEGE – from 4 to 7). Ensemble mean (line) and spread (error bars).

In a first step, the OASIS sequential exchange (namcouple SEQ mode) is preferred to the standard one: instead of exchanging all the fields before doing any interpolation calculation, the sequential exchange realizes field by field the sequence “get field – process oasis computations – put field”. So, OASIS does not need to wait for the slowest model to begin to processes the first interpolation. And interpolated fields are ready to be used by the slowest model as soon as it needs them. This optimization reduces the total elapsed time of our coupled simulation a 20% in the best case.

The second optimization consists on using the OASIS parallelism by field. This configuration called “Oasis pseudo parallelism” is implemented in OASIS3 version since Arnaud Caubel – Sébastien Masson – Jing-Jia Luo IPSL-JAMSTEC joint experiment on Earth Simulator supercomputer. Within this configuration, OASIS is launched several times with different namcouples, each namcouple describing a subset of one or several of the initial coupled fields needed. Several OASIS executables process a subset of the initial coupling fields: communications and interpolation calculations are done in parallel. This optimization (i.e., distributing OASIS on two processors, no extra gain observed with broader distribution) also reduces the total elapsed time of our coupled simulation by 18% in the best case.

Combining those two optimizations (Sequential mode + Pseudo parallelism), the cumulated gain compared to the non-optimized run varies between 15 and 25 %. Compared to the slowest model on a stand alone mode, the extra cost of coupling oscillates now between 0 and 25% depending on the load balance of the models. The highest coupling extra cost is obtained in the case of a load balanced configuration (4 processors for NEMO, 7 for ARPEGE): the difference between model durations is lower than the cost induced by the coupling.

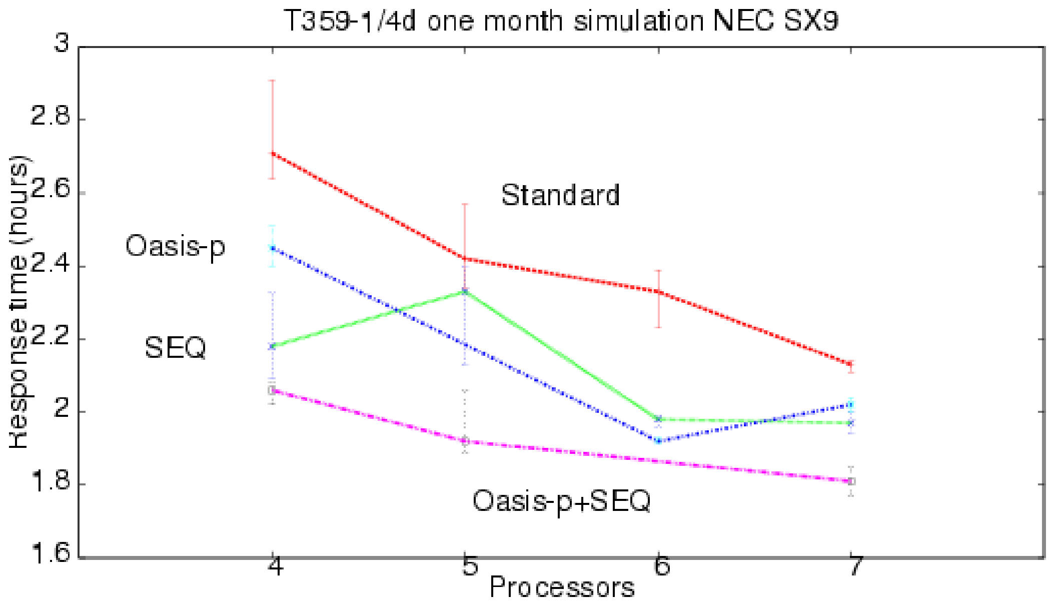


Figure 3.2: Compared performances without any optimization (red), with SEQ optimization (green, with parallelism by field (blue) and with combined SEQ+parallelism by field (pink). Ensemble mean (line) and spread (error bars).

To be totally sure that OASIS3 could handle model resolution higher than the present European most demanding configuration ones, we attempt to increase oceanic resolution, using the MERCATOR state-of-the-art operational model (1/12th degree, 50 vertical levels).

At such resolution, even during an OHC period, the machine load is so important that a limited number of tests is possible. A 9 member ensemble test of 2 simulated days has been processed, using 44 processors for NEMO and 4 for ARPEGE (3 nodes, half of the total amount of machine processors). Even if load balancing could not be totally reached, both Sequential mode + Pseudo parallelism optimizations help us to reduce the total simulation time from 8 to 4 hours, with only a 10 % additional time compared to the oceanic stand alone simulation.

Those experiments prove the OASIS3 capability to drive high end resolution coupled simulations on vector machines (even with experimental configurations using a 1/12th degree ocean) with reasonable additional time. Most of the time, when balance between model component duration cannot be reached, this additional time even could be nullified.

Acknowledgments:

Author is grateful to R. Bourdallé-Badie, O. Le Galloudec (MERCATOR) and M. Déqué (CNRM) for providing their NEMO and ARPEGE configurations. We also would like to thank S. Valcke (CERFACS) for fruitful discussions, M. Pithon (Météo-France), N. Monnier and I. d'Ast (CERFACS) for their help in code porting.

4. ARPEGE-NEMO porting and performance tests on DEISA-PRACE-GENCI platforms

Authors: E. Maisonnave[1], A. Caubel[2] and O. Jorba[2]

[1] Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, France (CERFACS)

[2] Institut Pierre Simon Laplace, France (IPSL)

[3] Barcelona Supercomputing Center, Spain (BSC)

4.1 Introduction

Built and optimized for vector platforms, could ARPEGE-NEMO climate model be used intensively on scalar machine like DEISA-PRACE-GENCI ones? This report describes the encountered difficulties during the model porting phase on SGI Altix ICE (CINES), IBM Blue Gene (IDRIS) and Marenostrum IBM JS21 (BSC) and details scalar/vector compared performances. Sensibility tests on horizontal resolution will help us to outline a first conclusion concerning scalar machine use for climate modelling.

4.2 Porting

CERFACS/Météo-France jointly developed climate model for AR5 IPCC exercise is built on ARPEGE-IFS atmospheric model. This model is derived from the operational version used by Météo-France for weather forecast and by ECMWF (European Centre for Medium Range Weather Forecast). We associate it with an ocean model (NEMO) using the OASIS coupler.

Several atmospheric model resolutions have been tested, from truncation 159 (T159, about 100km side cells) to truncation 359 (T359, about 50km side cells), with 30 vertical levels. Ocean is ½ degree broad.

ARPEGE version 5.1 has been used, NEMO version 3.1 and OASIS version 3.2 (tag prism_2-3).

We remind you that MPI MPMD functionality is mandatory for our OASIS coupler (for both V3 and V4). Several instances of each of three parallel executables which compose our climate model (atmosphere, ocean and coupler) are launched simultaneously (through mpirun or mpiexec command); OASIS addresses sub-communicators re-definition to allow concurrent internal parallelism and model parallelism.

Let's describe the major issues which occur during model install on the three machines we access through the French 2009 DARI supercomputing allocation and IS-ENES collaborations.

4.2.1 CINES SGI Altix

The cluster SGI Altix ICE 8200, JADE, is a scalar parallel supercomputer with a peak performance of 147 Tflop/s. JADE is composed of 25 racks. One of these racks is used to ensure the connection with the whole cluster (via 3 login nodes) and 24 computing racks.

This cluster consists of 12288 cores distributed on 1536 nodes (each node including 2 Intel Quad-Core E5472 processors). 30 GB of useful memory can be actually used on each node, i.e. over a total of 46 TB. The computing racks have access to a parallel file system Lustre with a total capacity of (509 TB).

As far as we know, before IPSL Grand Challenge experiment made with the whole IPSL LMDZ-NEMO climate model (see Report 2: *IPSL-ESM (LMDZOR-NEMO-OASIS3) porting and performance tests*) the oceanic component only was available on this platform. Consequently, compiler optimization upper to level O2 was not yet easily available on this machine.

For atmosphere model, several functionalities have been disabled, in particular the whole system of tracing routines (called “Mr Hook”). For future optimizations routine by routine, it will be necessary to identify the dysfunction and re-activate the tracing.

Problem with ARPEGE 2D parallelism in coupled mode

A greater problem appears using 2D intern parallelism on coupled mode. Actually, ARPEGE 2D parallelism improvements allow:

1. on routines where equation are solved on grid points, MPI domain decomposition is rectangular (following latitude but also longitude) and not only on latitude stripes, which allows, of course, a greater parallelism.
2. on routines where equation are solved on spectral mode, parallelism is done on spherical harmonics but also on vertical levels.

Parallelism skill on both grid points and spectral mode should be the same. That implies that decomposition is limited by the product of spectral decomposition times vertical level number, i.e. for a T159 resolution and 30 vertical levels, 3000 domains (about 7000 for a T359).

On coupled mode, it seems that previously implemented OASIS interface routines have not been updated to take into account the 2D parallelism. So, our study has been bounded by the 1D parallelism limitations. Domain decomposition won't exceed 110 for a T159 resolution, 250 for a T359. Those figures are quite far from the optimum measured with ARPEGE model on stand alone mode.

The most efficient optimization on coupling strategy has been deployed: OASIS parallelism by coupling field (up to 16 resources used) and sequential coupling mode. Improvements are in accordance with the one observed (at higher resolution) on NEC SX9 vector machine (see Report 3: *Development of a high resolution version of ARPEGE-NEMO climate model*): time spent on coupling routines related to communication or interpolation does not rise above the elapsed time difference between ocean and atmosphere model. At this resolution, it seems unnecessary to use an OASIS version insuring a greater scalability to the coupler (OASIS4).

Slowdown of ARPEGE internal time step in coupled mode

An issue appears comparing elapsed times of two atmosphere model configurations: on coupled mode and on stand alone mode. The difference (the coupled mode is slower) not only occurs at coupling time step, but at each model time step as well.

On coupled mode, each instance of the different models (and of the coupler) is mapped on a resource allocated according to an unusual algorithm (machine dependant and not changeable). Then, we cannot ensure that atmospheric processes won't share memory on a node with an other process (ocean or coupler), which can disturb communications between one model instances.

To limit those interaction risks, we launched 8 instances of our coupler, hoping that they will be mapped by the MPI launcher on 8 cores of the same node, leaving to instances of the other models a divisible by 8 numbers of resources.

According to this protocol, mean elapsed times for an atmosphere model time step (without IO, without convection calculations, without coupling), with NEMO on 64 cores (for coupled experiments only), are the following:

	ARPEGE on 16 resources	ARPEGE on 104 resources
ARPEGE stand-alone	1.05	0.195
CPL, OASIS on 1 resource	1.09	0.264
CPL, OASIS on 8 resources	1.1	0.264

Table 4.1: Elapsed time (seconds) for an ARPEGE T159 internal time step

Booking the node number of cores (8) for OASIS does not change the observed slowdown. And the slowdown increases with the number of cores allocated to ARPEGE (5% increase in elapse time with 16 cores, 25% with 104). But, without any information on mapping algorithm and their location, it won't be impossible to better investigate this issue.

Nevertheless, despite this unconstrained extra cost, best time performances with the T159 - ORCA ½ configuration approach 1.5 days to simulate 10 years, using about 400 cores, a quite reasonable figure compared to current configuration in use on vector machines at Météo-France (3 hours to simulate 10 years on 8 processors with a T127-ORCA1 configuration).

4.2.2 BSC MareNostrum IBM JS21

MareNostrum is a state-of-the-art High Performance Computing system held by BSC. More detailed information on the BSC MareNostrum HPC can be found in section 2.2.1.

Météo-France's climat model deployment on MareNostrum was straightforward due to availability of NEMO and OASIS codes (previously installed by Arnaud Caubel (IPSL) and David Vicente (BSC)). A simple modification on OASIS code was performed to trace model components separated performances and a script provided to collect and analyse the results. Description of this counter is made on section 3.2 *Experimental design*, on Report 3 *Development of a high resolution version of ARPEGE-NEMO climate model for OASIS3 testing purpose*.

ARPEGE porting was neither problematic, due to previous compiling with IBM XL FORTRAN on other IBM platforms.

4.2.3 IDRIS IBM Blue Gene/P

The IBM Blue Gene/P system of IDRIS (babel) has 10 racks, each one containing 1024 compute nodes. A compute node has 4 computing cores running at 850MHz and 2GB of memory. The total theoretical peak performance is 139Tflops (3.4Gflops by core).

Each rack is divided in two midplanes, each one containing 512 compute nodes (2048 cores). The compute nodes are grouped by 64 (“pset”) and each group has 1 I/O node. Therefore, it is only possible to allocate partitions (group of compute nodes) by multiple of 64

(256 cores).

The machine has access to parallel filesystems with a total capacity of nearly 800TB.

Previously used on IBM Blue Gene/L and JS21 Mare Nostrum machines, both ocean and atmosphere compilations were particularly easy going.

A fatal complication happened during OASIS execution phase: Due to some characteristic of BG/P software suite, OASIS version 3 cannot be used without major code changes.

Actually, MPI MPMD launching mode implemented on this machine has two kinds of restrictions:

1. an executable must be launched on an integer number of “pset”
2. MPI ranks given to instances of a same executable are not contiguous.

Our 3rd version of OASIS has been late adapted to a parallel use. MPI instance number could not actually exceed the number of coupling field (15, in our case), which was not compatible with the first constraint (minimum of 256 instances for one executable). A recent improvement enables now OASIS to run without coupling fields: these “dummy” OASIS instances could be added to the useful ones to reach the mandatory figure.

Secondly, OASIS3 sub-communicator definition associated to coupled components imposes that MPI ranks of coupler instances should begin at zero, and should increase monotonically (rank 0,1,2,3,4,5,6,7 for 8 OASIS instances). That's not compatible either with the second constraint.

For the first time, we reach here a limitation of our pseudo-parallel OASIS3 coupler. To complete our study on this machine, and to have a chance to process climate simulations on this IBM Blue Gene/P machine, it is mandatory to develop the OASIS4 version of our coupled system. This task began during the 2009 OASIS user support at IPSL (IS-ENES WP4), designing the new interface of NEMO model.

Generally speaking, we also might wonder about the compatibility of our approach, based on separate executable coupling and MPI MPMD mode, with the observed standardization of supercomputers. We noticed that, for the moment, supercomputer users of others community poorly take advantage of this mode. This is clearly a non standard use of the MPI library. Should this technique become problematic on standardized machine such as massively parallel supercomputers?

4.3 Performances of ARPEGE atmospheric component

Taking advantage of LMDZ' and ARPEGE's presence on the same supercomputer, some common analysis have been performed with “PARAVER” BSC toolkit (see Report 2: *IPSL-ESM (LMDZOR-NEMO-OASIS3) porting and performance tests*).

	LMDZ 1 task/node	**LMDZ 4 tasks/node**	**ARPEGE 4 tasks/node**
Instructions per cycle	0.62	0.47	0.33
Parallel efficiency	0.78	0.87	0.92

Table 4.2: Common LMDZ-ARPEGE analysis performances on Mare Nostrum (BSC)

If ARPEGE scalability exceed LMDZ' one, ARPEGE exhibits a slower number of instructions per cycle, which could imply a higher number of memory access conflicts. Parallel efficiency is slightly the same. All those figures could suggest that ARPEGE's higher scalability should be mainly originated in its slower amount of communications.

Scalability tests have been processed with atmospheric model using two kind of resolution: standard (T159, used by several laboratories within AR5 IPCC exercise) and high (T359, used for short term forecasts). The high resolution version could be used for climate research purposes, but still on experimental experiments, requiring an elevate amount of computing resources.

Please notice that, on every machine, I/O routines have been disabled, due to performance reasons but also for 2D parallelism mode considerations (see section 4.2.1 *CINES SGI Altix*).

On IBM BG/P, « DUAL » mode has been preferred (two physical cores out of 4 are effectively used). On SGI Altix, every 8 cores of a node are used. The same on IBM JS21, where 4 physical cores out of 4 are effectively used.

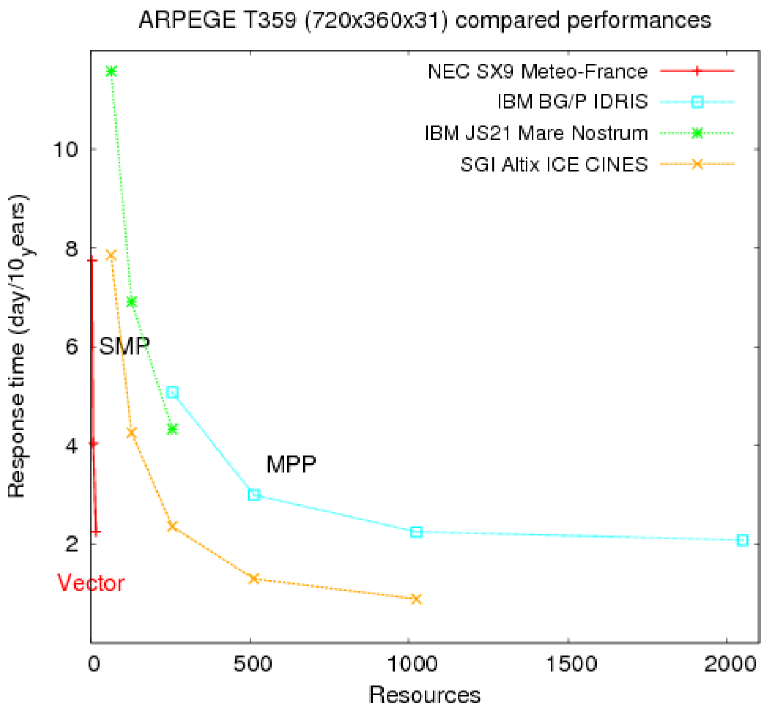


Figure 4.1: Compared performances (elapsed time) on several platforms for ARPEGE model (T159 resolution)

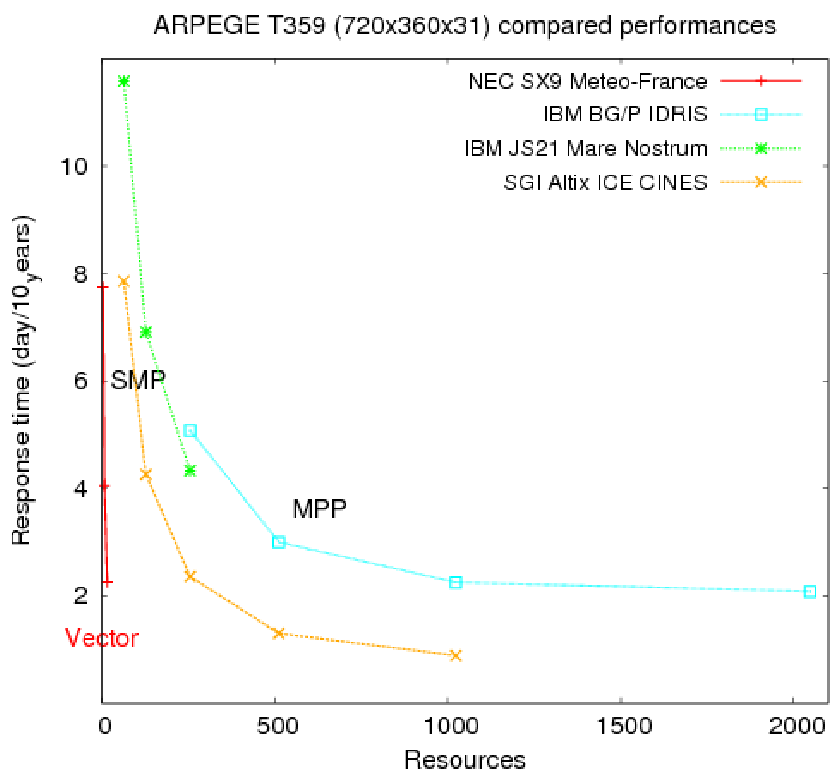


Figure 4.2: Compared performances (elapsed time) on several platforms for ARPEGE model (T359 resolution)

Those results suggest that time performances of the standard resolution ARPEGE model (without IO) on those 3 scalar platforms are compatible with a production use (for both short term and long term experimental needs). It's the same for the “frontier” T359 resolution, considering that a hundred year of simulation can be processed during less than 15 days (using 500 cores on SGI Altix machine).

For a production use, those good results must be confirmed and important improvements must be undertaken:

- additional developments for IO enabling
- NEMO scalability (done after this report writing) and IO improvements (IO server, currently under development for MPP compatibility)
- ARPEGE coupling interface rewriting to allow 2D parallelism compatibility

This present study on compared performances of ARPEGE's different resolution configurations on highly parallel scalar machines convinced us to suggest a choice on model definition for years to come.

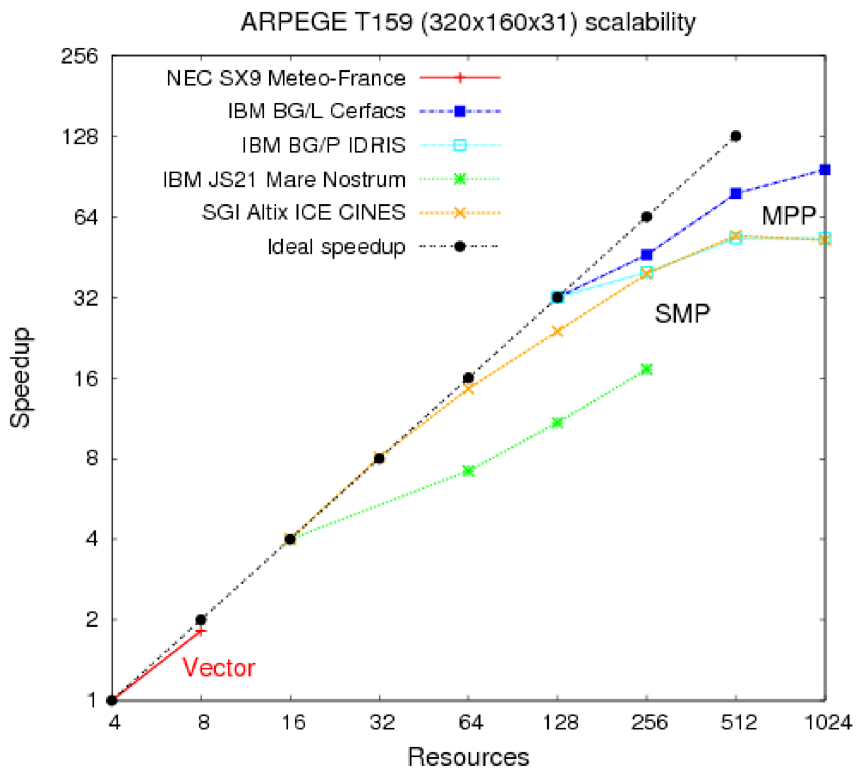


Figure 4.3: Compared scalabilities on several platforms for ARPEGE model (T159 resolution)

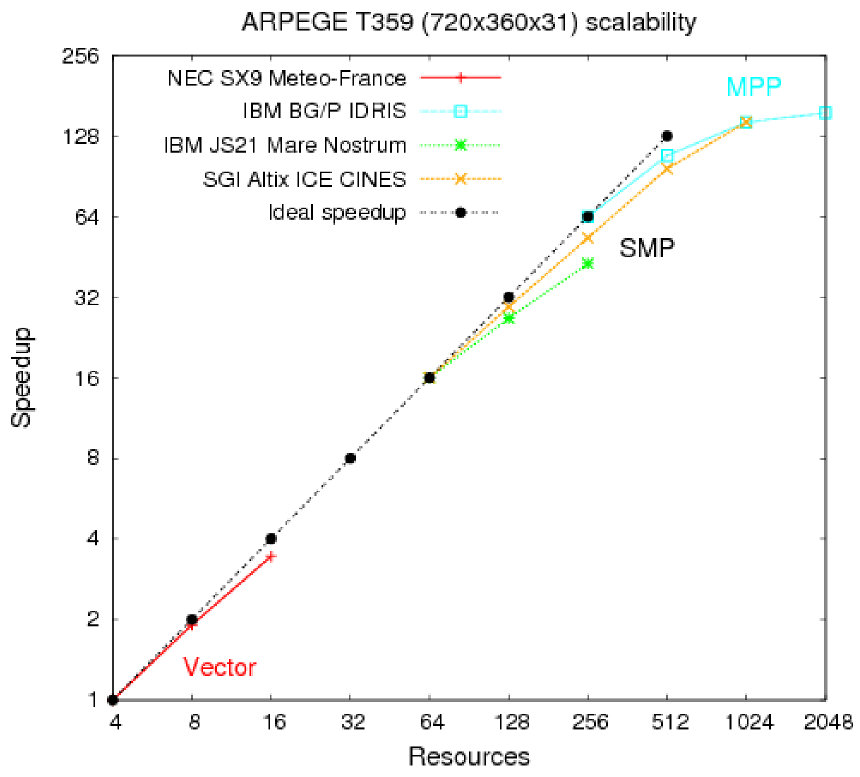


Figure 4.4: Compared scalabilities on several platforms for ARPEGE model (T359 resolution)

As we can see on these figures, speed-up keeps acceptable values up to 500 cores, using high resolution version of the model, whereas it weakens over 100 cores at standard resolution.

This result cannot be extrapolated unlimitedly but it suggests that “weak scaling” could offer a good solution to be able to test higher decompositions (up to 10.000 to 100.000 domains) with higher resolution and begin to investigate issues that we sooner or later will encounter (like fault tolerance compliance).

Our model scalability increasing with resolution, this is our “frontier” configuration that might better take advantage of future massively parallel platforms.

Acknowledgments:

Authors are grateful to support teams of computing centres (IDRIS, CINES, BSC, CERFACS) and vendors (IBM), particularly to Isabelle Dupays, Thierry Goldman, Pascal Vezolle, Mathieu Cloirec et Isabelle d'Ast.

5. Porting and performance analysis of EC-Earth system on MareNostrum Supercomputer

Authors: F. Martínez[1], D. Vicente[1], O. Jorba[1]and M. Val Martin[1]

[1] Barcelona Supercomputing Center (BSC)

5.1 Introduction

5.1.1 EC-Earth System Model and BSC MareNostrum Supercomputer

The European Community Earth system model (EC-Earth) is an Earth System Model (ESM) that was developed by a number of European National Weather Services and university groups. It is based on the seasonal prediction system of European Centre for Medium-Range Weather Forecasts (ECMWF) [Hazelenger et al, BAMS].

EC-Earth is under current development and consists of two main components: an atmosphere, chemistry, land and vegetation model and an ocean and sea-ice model. Each of these two main components contains various sub-components, which represent physical processes, and climate-related biological and geochemical processes. These models communicate with each other through a coupler. The used programming languages are FORTRAN and C [Brandt, 2010]. The final software architecture of EC-Earth will consist of five main software codes:

1. **Coupler:** The coupler software used is OASIS3, which manages a number of operations related with a simulation with two o more interacting models. The interacting models are the atmosphere model together with a land and vegetation module, an atmospheric chemistry model and an ocean and sea-ice model.
2. **Atmosphere model:** The atmosphere model of EC-Earth is based on cycle 31r1 of the Integrated Forecasting System (IFS) of ECMWF. The basic configuration contains 62 levels in the vertical, with a model top at 5hPa (~37 km). The dynamical part of the model uses a spectral transform approach, with a present horizontal resolution of T159. The physical parametrization schemes of the model (including clouds, rain, radiation, turbulence and land surface processes) are all calculated on a reduced N80 Gaussian grid, which corresponds to a 1.125 degrees spacing (125 km). The atmospheric model uses a two time-level semi-Langrangian scheme for its dynamics with a 1-hour time step.
3. **Ocean and sea-ice model:** The ocean and sea-ice model is based on version 2 of NEMO (Nucleus for European Modelling of the Ocean). The ocean general circulation model (OGCM) is OPA (Océan Parallélisé). OPA is a primitive equation model which is numerically solved on a global ocean curvilinear grid known as ORCA. ORCA1 has a resolution of 1 degree and the southern pole grid is refined to resolve the circumpolar currents. The vertical grid has 31 levels (the 31st level is below the ocean bottom) with variable layer depth and a constant 10 m step in the top 150m. The Louvain-la-Neuve sea-Ice Model (LIM) is a thermodynamic-dynamic sea-ice model directly coupled with OPA.
4. **Atmospheric chemistry model:** The global chemistry model of EC-Earth, is called TM5. It describes the atmospheric chemistry and transport of reactive or inert tracers It can be run on a global spatial resolution of 6x4 or 3x2 degrees, with the option to increase the resolution to 1x1 degrees over specific regions, e.g. over Europe, the United States or Asia. It has been included as an online module that evaluates the transport and concentrations of reactive gases and various aerosol types for the

meteorological conditions simulated by IFS. The concentrations of the simulated greenhouse gases and the concentrations and relevant optical properties of the aerosols can be fed back to calculate the associated direct and indirect radiative forcings in IFS. Exchange between TM5 and IFS currently takes place on a 3- or 6-hourly basis.

5. **Land and vegetation module:** The land-vegetation model is the hydrological extension of the Tiled ECMWF Surface Scheme for Exchange processes over Land (HTESSEL) and it is part of the atmosphere model.

The EC-Earth ESM has been ported successfully over different high performance computing platforms (e.g., IBM P6 AIX, CRAY XT-5, Intel-based Linux Clusters, SGI Altix, MareNostrum) at different sites in Europe (e.g., KNMI, ICHEC, ECMWF). The current version of EC-Earth consists of three models: IFS, OASIS and NEMO. The development of the different model versions was as follows:

- Version 0: Uncoupled model IFS CY31R1
- Version 1: Coupled model (IFS CY31R1– OASIS3 v2.5 – NEMO2/LIM2)
- Version 2: Coupled model (IFS CY31R1– OASIS3 v2.5 – NEMO2/LIM2), with local ECMWF modifications and EC-Earth developments for CMIP5
- Version 3: Coupled model (IFS CY33 – OASIS3 v2.5 –NEMO3/LIM3)

The MareNostrum HPC is located at the Barcelona Supercomputing Center (BSC). A more detailed description on the BSC MareNostrum HPC can be found in section 2.2.1.

5.2 Objectives

The main objective of this work is to identify and document the issues related with the portability and performance of the EC-Earth model on the MareNostrum supercomputer. In order to meet the standards for future HPC architectures (e.g., within the PRACE infrastructure), it is important to understand the current performance of ESMs on state-of-the-art computing systems. In this sense, we focus on the porting and performance analysis of EC-Earth model on the MareNostrum supercomputer, a DEISA2 node.

In this report, we describe the efforts done to port the coupled EC-Earth model on MareNostrum and the preliminary optimization tasks undertaken to improve the default performance of the modeling system. A description of the execution characteristics of the system is presented and results from a scalability study are discussed. Finally, we present initial analysis of raw performance traces using the Paraver analysis software tool developed at BSC (http://www.bsc.es/plantillaA.php?cat_id=485) and conclusions for future work.

5.3 Description of execution and evaluation tests

5.3.1 Porting

Three versions of EC-Earth have been compiled and run in MareNostrum: version 2.0, 2.1 and 2.2. Detailed instructions for compiling and running the model in its version 2.0 are found in Stefanescu, 2008.

To compile EC-Earth (versions 2.0, 2.1 and 2.2) at MareNostrum some modifications on the code were need. First, we included the flag -qextname for the multiple configuration files and routines that required extra underscoring. The most optimized compiler in MareNostrum is IBM XL compilers since MareNostrum is an IBM machine with PPC970 processors. This type of compilers does not include by default the underscoring in the Fortran routines, which it was required by the C code that calls to Fortran routines in the original EC-Earth code. Second, changes in the source code were required to adapt the program to our architecture, so the IBM AIX structure was selected as base. This approach required some tuning to match the architecture IBM PPC LINUX running at BSC.

In addition to the changes described above, a particular compilation for IFS was needed for versions 2.1 and 2.2 as a result of an incompatibility of the IFS software and the compiler XLF version 12.1. After extensive testing, a two step compilation approach was used to compile IFS within EC-Earth using XLF version 10.1 and 12.1 compilers. Compiling all IFS with XLF v10.1 was not feasible as other modules within EC-Earth were only available for the new XLF v12.1 compiler.

A platform intercomparison using EC-Earth version 2.1 and a simplified 10-year simulation run showed that EC-Earth ports and performs well in different architectures of European supercomputing centers including MareNostrum.

5.3.2 Execution

EC-Earth is executed in parallel using MPI and the default setting of 16 CPUs for NEMO and 1 CPU for OASIS. For NEMO, a default number of 16 CPUs was established by the EC-Earth community. For OASIS, one CPU is used as EC-Earth does not use a parallelized version of this coupler. Table 5.1 summarizes the requirements and resources needed to execute EC-Earth in our platform.

MareNostrum uses a batch processing support, so all jobs must be run through it. The batch system used in MareNostrum is a combination of two softwares: 1) SLURM, developed at Lawrence Livermore National Laboratory and designed for large clusters, which works as a resource manager and 2) MOAB, developed at Cluster Resources company, which works as a job scheduler. The user then needs to specify the number of CPUs allocated for each task. This is useful for hybrid MPI+OpenMP applications, in which each process spawns a number of threads. The number of CPUs per task must be between 1 and 4, since each node has 4 CPUs (one for each thread). It is important to note that OpenMP settings are globally defined for a whole run. It is also possible to define the number of tasks allocated in each node. When an application uses more than 1.7GB of memory per process, it is not possible to have 4 processes in the same node because of an 8GB memory limit. Due to this special configuration, the submitting batch script to send an EC-Earth run to the MareNostrum queues is different to other platforms (see Table 5.1 for details).

Platform	
Execution platform	MareNostrum
Requirements	**ksh** - korn shell (ksh93) **perl** - used extensively with generic Makefile to build source libraries **mpi1** - required for parallel execution **Fortran 95** - a fortran 95 compiler that supports an auto-double (or -r8) capability. OASIS3 requires Cray pointers. **OpenMP** - if OpenMP is used then the MPI implementation is required to be thread-safe **netCDF** - the netCDF library for I/O of all NEMO and OASIS3
Libraries	**Oasis**: netcdf **Nemo**: netcdf **IFS:**

	• blas - a standard (“off the web”) blas library • dummy - stubs for routines that are never called • ec- a small number of ecmwf utility routines • emos - library of data manipulation routines, like bufrdc/ bufr • (observation l e routines), gribex/ grib (spectral/grid point • fields) and pbio/ simple C based I/O routines • IFS - contains the main IFS source library, has many subdirectories • IFSaux - contains IFS utility routines (in particular MPL * interface to MPI) • lapack - public domain source of LAPACK routines (see also blas) • prepdata - data preparation routines • surf - surface package • trans - IFS transform routines/data distribution
CPUs Used for each component	**IFS:** 64 (NPRTRW =32 and NPRTRV =2) NPROC: Total number of CPUs (NPRTV x NPRTRW) NPRTRW: Number of CPUs used during transform phase in wave space (max 47) NPRTRV: Number of CPUs used during transform phase in vertical direction (max 62) **OASIS:** 1 **NEMO:** 16 (NEMOPROCX=4 and NEMOPROCY=4)

Submit job command	NEMO_nproc=$((NEMO_nprocX*NEMO_nprocY))
	minNEMO_nproc=$((IFS_nproc+1))
	maxNEMO_nproc=$((IFS_nproc+NEMO_nproc))
	total_nproc=$((maxNEMO_nproc+1))
	echo "0 ${OASIS3_exe}" > silly.conf
	echo "1-$IFS_nproc ${IFS_exe} -v ecmwf -e $EXPVER" >> silly.conf
	echo "$minNEMO_nproc-$maxNEMO_nproc ${NEMO_exe}" >> silly.conf
	srun -n$total_nproc -l --multi-prog silly.conf

Table 5.1: Summary of requirements and resources needed to run EC-Earth.

5.3.3 Optimization

The optimization process mainly focused on adapting the compilation flags in order to improve the efficiency of the code (e.g., inlining of specific functions, test different levels of optimizations and libraries, etc). In addition, we checked the performance of the I/O in the two filesystems available in MareNostrum: 1) /scratch/, which is the local filesystem only accessible from each node and with 36 GB, and 2) /gpfs/, which is globally available from any node and with more than 100 TB of storage space. We chose to run EC-Earth within gpfs/ because the use of the local filesystem did not improve the I/O behaviour, and scratch/ was limited in space and output files were more difficult to recover.

5.3.4 Scalability

To understand the performance of EC-Earth in MareNostrum, we did a scalability test using EC-Earth version 2.2. The experiment consisted of running 1-month simulations for different numbers of CPUs for IFS (i.e., 4, 8, 16, 32, 36, 64, 96, 128 and 256). In this study, we kept constant setting of 16 CPUs for NEMO and 1 CPU for OASIS. We considered the number of

IFS CPUs as multiple of four because MareNostrum has four CPUs per node and to avoid performance issues due to share memory resources with other applications. A constant number of 16 CPUs was considered for NEMO since that was the default setting established by the EC-Earth community for the Coupled Model Intercomparison Project (CMIP-5).

Figure 5.1 shows the speedup and the scalability efficiency of EC-Earth for different number of IFS CPUs, and Table 5.2 summarizes the execution times. Only results from 4 to 128 IFS CPUs are shown because running EC-Earth with 256 IFS CPUs was not feasible due to MPI communication problems (only 2 out of 10 tests were successful). The relative speedup for each number of CPUs is calculated as the ratio of execution time of the base number of CPUs (in this case, 4 IFS CPUs, i.e., a total of 24 CPUs) and the execution time with X CPUs; efficiency is the relative speedup multiply by the ratio of the CPUs (i.e., 24 over X). It is important to note that the standard metric speedup is the ratio over the run with only one CPU. We used the metric speedup ratio in this analysis as our objective was to determine the optimum number of IFS CPUs to efficiently run EC-Earth. These results show that the efficiency of EC-Earth in MareNostrum decreases when a large number (> 64) of IFS CPUs are used, and that the optimum number of IFS CPUs to run EC-Earth in MareNostrum is 64. It is important to note that the efficiency values are above one, which indicates an issue on the EC-Earth performance when using a low number of IFS CPUs (<16). In addition, we tested the EC-Earth performance for different balances between *NPRTRW* and *NPRTRV* and found no significant performance variation.

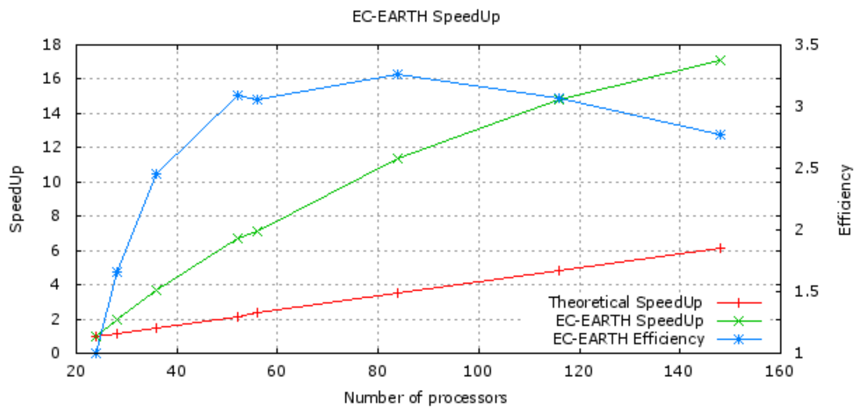


Figure 5.1: Speedup and scalability efficiency of EC-Earth runs from 24 to 148 CPUs. Numbers reported are the average of 10 runs (excluding the lower and the upper outlier to avoid MareNostrum instability performances).

Number of CPUs	Execution Time
24 (1 + 16 + 4)	13:53:58
28 (1 + 16 + 8)	07:13:39
36 (1 + 16 + 16)	03:48:08
52 (1 + 16 + 32)	02:05:11
56 (1 + 16 + 36)	01:57:49
84 (1 + 16 + 64)	01:13:35
116 (1 + 16 + 96)	00:56:35
148 (1 + 16 + 128)	00:49:00

Table 5.2: Execution time for each EC-Earth run. Numbers reported are the average of 10 runs (excluding the lower and the upper outlier to avoid MareNostrum instability performances). Total number of CPUs is rounded to the closest multiple of four.

Table 5.3 shows a summary of computation times within EC-Earth considering coupling and no-coupling times. The computation time is the addition of the time from the log for all time step, the time without coupling is estimated as the product of number of time steps by the mean time of the time steps without coupling stage and the coupling percentage is the fraction of time that the code uses for coupling, based on the coupling and non-coupling times. We find that the coupling percentage increases with the number of CPUs due to a decrease in computation time. From the log files (not shown), we find that there is a significant larger time for processing every 3 hours, likely as a result of this coupling.

Number of CPUs	Computation time	Coupling percentage	Time without coupling
24 (1 + 16 + 4)	13:53:23	13,27%	12:02:50
28 (1 + 16 + 8)	07:11:45	13,73%	06:12:27
36 (1 + 16 + 16)	03:46:52	14,56%	03:13:51
52 (1 + 16 + 32)	02:03:46	22,01%	01:36:25
56 (1 + 16 + 36)	01:57:24	25,16%	01:27:51
84 (1 + 16 + 64)	01:11:50	29,14%	00:50:54
116 (1 + 16 + 96)	00:55:41	34,21%	00:36:38
148 (1 + 16 + 128)	00:47:22	39,20%	00:28:48

Table 5.3: Example of distribution of computing and coupling time extracted from a single EC-Earth run. The total number of CPUs is rounded to the closest multiple of four.

5.3.5 Traces

We used Paraver to further test the performance of EC-Earth in MareNostrum. Paraver is an open source performance visualization and analysis tool developed in BSC. Figure 5.2 shows an example of one of the Paraver visualization modes: colors indicate the CPU states (e.g., running, waiting for communication, synchronization, group communication, communication send, I/O), the xaxis represents time, and the yaxis represents the different CPUs (e.g., 1 for OASIS, 4 for IFS and 16 for NEMO in our example).

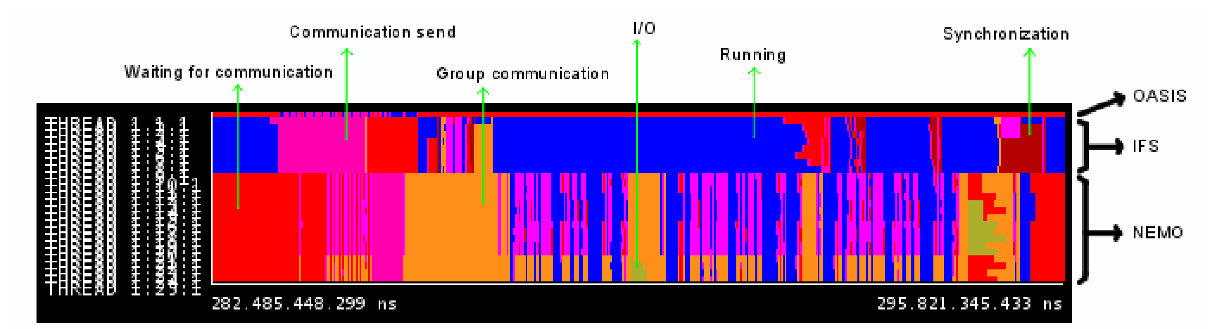


Figure 5.2: Example of a Paraver graphical view. OASIS, IFS and NEMO are run with 1, 8 and 16 CPUs, respectively. .

In addition from visualizing the CPU behaviour, performance statistics can be extracted from Paraver. We analyzed some of these parameters: the “load balance” is the average percentage of time each thread is executing over the maximum of that percentage (one for an ideal perfect balanced code and one/CPUs# for a totally unbalanced code); the “communication factor” is the maximum of the percentages of time any thread is executing, the “code replication” reflects the increased work the model will have to do, due to non-parallelized code repeated over several CPUs and the cost of the computations needed to perform the communications, i.e., the total number of instructions of that executions over the number of instructions executed for the 24 CPUs run; the “average instruction per cycle (IPC)” describes the internal behaviour of the code; the “speedup” is the execution time of the base number of CPUs (in this case, 24), over the execution time with X CPUs; the “efficiency” is the ratio of speedup over the ratio of the CPUs.

For a reference, the range obtained from load balance, communication factor, code replication and average IPC parameters gives an idea of the parallelization performance of the code in a specific aspect (> 0.9 well, 0.8-0.9 fairly well and <0.8 can be improved). It is important to note that values below 0.8 do not mean that there is a problem within the code, as this regular performance might be inherent to the model.

As a preliminary test, we created traces for 1-day simulation runs. Figure 5.3 shows results for the tests performed with 16, 32 and 64 IFS CPUs. This test shows the existence of a load imbalance between IFS and NEMO (see red areas in NEMO compared to IFS, i.e., NEMO is waiting for communication while IFS is still processing). However, this load imbalance improves as the number of IFS CPUs increases. In addition, we find that somehow there is a serial processing at the startup and some other periods (see highlighted regions in green), and that there is a significant larger time for processing every 3 hours due to the coupling (not shown).

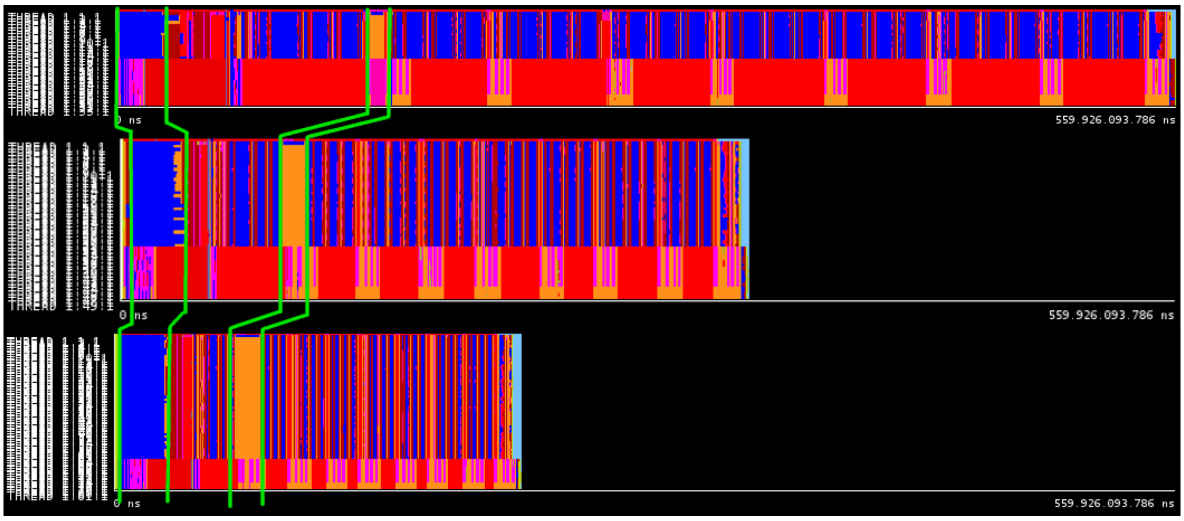


Figure 5.3: Trace visualization of a 24 hour EC-Earth simulation for 16 (top), 32 (middle) and 64 (bottom) IFS CPUs. NEMO and OASIS were run with 16 and 1 CPUs, respectively. Colors indicate running (blue), waiting for communication (light red), synchronization (dark red), group communication (orange), communication send (pink) and I/O (green). Green vertical lines indicate the initialization part of the trace that does not scale.

Figure 5.4 shows a comparison of a 3-hour run, including one coupling step, for EC-Earth using 16, 64 and 128 IFS CPUs (please, note the time axis scale is different in each plot). This Paraver analysis shows clearly that OASIS acts as a bottleneck and is always an important limiting factor independently of the number of IFS CPUs used because during coupling both IFS and NEMO are waiting for communication. Based on the Paraver analysis, we find that the coupling time takes about 2 seconds, confirming that the coupling time calculated in Table 5.2 corresponds in fact to the coupling stage.

In addition, we observed the existence of an extreme load imbalance between IFS and NEMO on the 16 IFS CPUS and a large imbalance for 64 IFS CPUs, indicating that IFS acts as a bottleneck and another important limiting factor in EC-Earth. The load imbalance improves with 128 IFS CPUs and IFS and NEMO can both act as limiting factor depending on the time step.

Overall, this analysis shows that IFS spends most of the time running, whereas NEMO spends an important fraction of time doing data transfer. This implies that, at MareNostrum, NEMO may not scale well beyond 16 CPUs. Previous studies showed the significant scalability of NEMO using MPI-OpenMP and recommend running NEMO using this parallel technique. However, we use only MPI parallelization for NEMO at MareNostrum because IFS is run with MPI. MareNostrum is a distributed memory system and does not support a hybrid parallel execution, i.e., one software running with MPI parallelization and another with OpenMP+MPI within the same submit job.

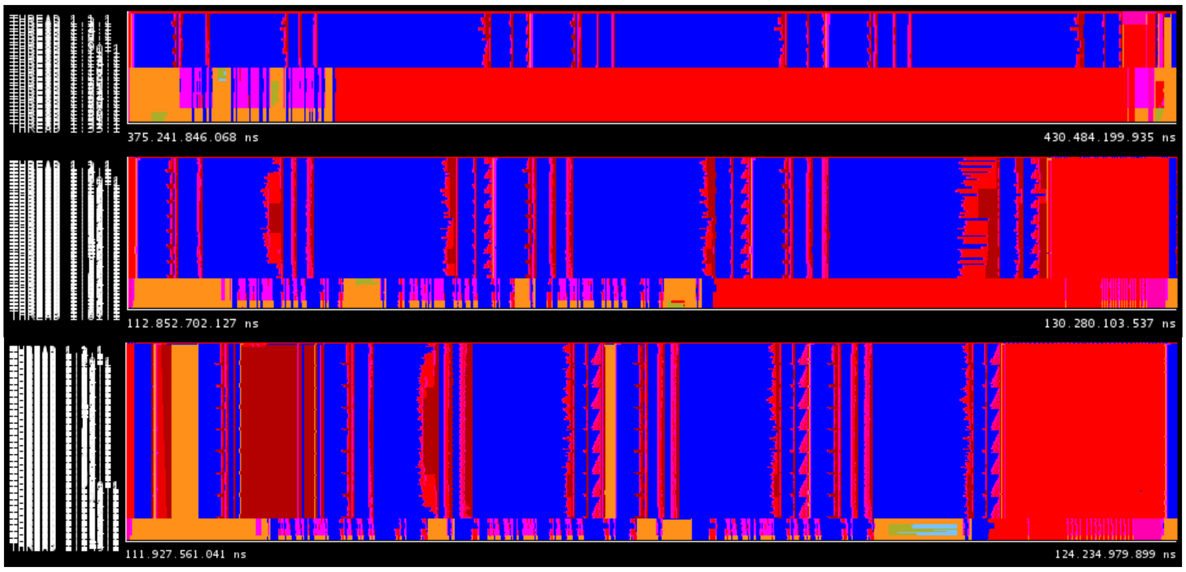


Figure 5.4: Example of a 3-hr trace visualization EC-Earth, showing a close-up view of the coupler communication for 16 (top), 64 (middle) and 128 (bottom) IFS CPUs. NEMO and OASIS were run with 16 and 1 CPUs, respectively. Colors indicate running (blue), waiting for communication (light red), synchronization (dark red), group communication (orange), communication send (pink) and I/O (green).

Table 5.3 summarizes the statistical parameters calculated from the Paraver analysis. This table shows that load balance tends to improve as the number of IFS CPUs increases. Communication performance decreases with the number of IFS CPUs, as a result of a reduction on the percentage of time that is used for calculations while the amount of communication remains the same. Similar behaviour is observed for code replication due to an increase in the calculations needed to distribute the workload and to non-parallelization of parts of the code. Not surprisingly, the average IPC improves as the number of IFS CPUs increases. Based on a further analysis of the average IPC from Paraver (not shown), we find that IFS has a significant larger IPC than NEMO and OASIS. Similarly to results found in section 1.3.4, the Paraver analysis confirms that the efficiency of EC-Earth in MareNostrum decreases when a large number (> 64) of IFS CPUs is used, and that the optimum number of IFS CPUs to run EC-Earth in MareNostrum is 64.

# of CPUs	**24**	**28**	**36**	**52**	**56**	**84**	**116**	**148**
Load balance	0,21	0,36	0,54	0,71	0,73	0,84	0,84	0,64
Communications	0,97	0,95	0,91	0,84	0,84	0,76	0,72	0,62
Code replication	1	0,82	0,92	0,69	0,61	0,57	0,55	0,43
Avg. IPC	0,31	0,37	0,38	0,5	0,59	0,63	0,64	0,66
Speedup	1	1,92	3,63	6,65	7,39	11,63	15,22	12,61
Efficiency	1	1,65	2,42	3,07	3,17	3,32	3,15	1,22

Table 5.3: Statistic metrics obtained with Paraver excluding the initialization stage (i.e., 21-hour run). Total number of CPUs is reported to the closest multiple of four.

5.4 Summary and Conclusions

The EC-Earth ESM was successfully ported to the BSC MareNostrum supercomputer, a system based on IBM PowerPC 970MP processors and run under a Linux Suse distribution. We detected a compilation incompatibility of IFS in EC-Earth v 2.1 and 2.2 to the compiler XLF version 12.1. This incompatibility was solved by a two step compilation approach using the XLF version 10.1 and 12.1 compilers.

The EC-Earth performance was analyzed with respect to scalability and trace analysis with the Paraver software. The performance analysis was done using the standard configuration of EC-Earth version 2.2, which uses 1 CPU for OASIS3 and 16 CPUs for NEMO, and modifying the number of CPUs allocated for the IFS model.

Our analysis showed that EC-Earth performs fairly well in MareNostrum, and that the efficient number of IFS CPUs is 64. Running EC-Earth with a larger number of IFS CPUs (>128) is not feasible at the moment since some issues exists with the IFS-NEMO balance and MPI communications. We detected a negligible load imbalance within IFS, which is typical in models with complex physical calculations, and an important performance loss of EC-Earth in the coupling stage, as OASIS acts as a bottleneck since the serial version of OASIS is used.

Acknowledgments:

We wish to thank Xavier Abellán from the BSC Operations team, and Pedro Jiménez from Universidad de Murcia for their help porting and running EC-Earth in Marenostrum, and Judit Giménez of Computer Sciences Department of BSC for her help on the analysis of Paraver traces. We appreciate Simona Stefanescu from ECMWF and Tido Semmler from Met Eireann for technical help with the EC-Earth scripts.

6. Performance analysis and porting of the CMCC-MED model

Authors: I. Epicoco[1], S. Mocavero[2], G. Aloisio[1,2],

[1] University of Salento, Lecce, Italy

[2] Euro-Mediterranean Center for Climate Change (CMCC)

6.1 CMCC-MED model description

The CMCC-MED model, developed under the framework of the EU CIRCE Project (Climate Change and Impact Research: the Mediterranean Environment), provides the possibility to accurately assess the role and feedbacks of the Mediterranean Sea in the global climate system. CMCC-MED is a global coupled ocean-atmosphere general circulation model (AOGCM) coupled with a high-resolution model of the Mediterranean Sea. The atmospheric model component (ECHAM5) has a horizontal resolution of about 80 Km with 31 vertical levels, the global ocean model (OPA8.2) has horizontal resolution of about 2° with an equatorial refinement (0.5°) and with 31 vertical levels, the Mediterranean Sea model (NEMO in the MFS implementation) has horizontal resolution of 1/16° (~7 Km) and 72 vertical levels. The communication between the atmospheric model and the ocean models is performed through the CMCC parallel version of OASIS3 coupler, and the exchange of SST, surface momentum, heat, and water fluxes occurs every 2h40m. The total number of fields exchanged through OASIS3 is 35. The global ocean-Mediterranean connection occurs through the exchange of dynamical and tracer fields via simple input/output operations. In particular, horizontal velocities, tracers and sea level are transferred from the global ocean to the Mediterranean model through the open boundaries in the Atlantic box. Similarly, vertical profiles of temperature, salinity and horizontal velocities at Gibraltar Strait are transferred from the regional Mediterranean model to the global ocean. The ocean-to-ocean exchange occurs with a daily frequency, with the exchanged variables being averaged over the daily time-window.

6.1.1 Configuration

In Table 6.2 the compilation keys for each component model are reported.

Model Name	Compilation keys
NEMO - Mediterranean Sea	key_dynspg_flt key_ldfslp key_zdfric key_dtasal key_dtatem key_vectopt_loop key_vectopt_memory key_oasis3 key_coupled key_flx_circe key_obc key_qb5 key_mfs key_cpl_discharge_echam5 key_cpl_ocevel key_mpp_mpi key_cpl_rootexchg key_useexchg
ECHAM5 - Atmospheric	__cpl_opa_lim __prism __CLIM_Box __grids_writing __cpl_maskvalue __cpl_wind_stress

Model Name	Compilation keys
OPA8.2 - Ocean Global	key_coupled key_coupled_prism key_coupled_echam5 key_coupled_echam5_intB key_orca_r2 key_ice_lln key_lim_fdd key_freesurf_cstvol key_zdftke key_flxqsr key_trahdfiso key_trahdfcoef2d key_dynhdfcoef3d key_trahdfeiv key_convevd key_temdta key_saldta key_coupled_surf_current key_saldta_monthly key_diaznl key_diahth key_monotasking

Table 6.2: Compilation keys for models

Each component model is used with the spatial and temporal resolutions shown in Table 6.3, while the coupler OASIS3 has been configured as in Table 6.4.

	OPA8.2	ECHAM5	NEMO
time step	4800s	240s	600s
grid points	182x149	480x240	871x253
vertical levels	31	31	72

Table 6.3: Spatial and temporal resolution of the component models

OASIS3 configuration			
Coupling period	9600s		
Total number of fields to be transformed	35		
	Number of fields exported to	Number of fields imported from	LAG
OPA8.2	17	6	4800s
ECHAM5	9	26	240s
NEMO	9	3	600s

Table 6.4: OASIS3 configuration

A detailed view of the transformations performed by OASIS3 on the exchanged fields is given in Table 6.5.

	Time	Pre-proc Transf.			Interp Transf.				Cook stage		Post-proc
Field ID	LOCTRANS	MASK	EXTRAP	INVERT	DISTWGT	CONSERV	BILINEAR	BICUBIC	CONSERV	BLASNEW	REVERSE
1	✓	✓	✓		✓					✓	✓
2	✓					✓					✓
3	✓	✓	✓				✓			✓	✓
4	✓	✓	✓				✓				✓
5		✓	✓				✓				✓
6	✓	✓	✓				✓				✓
7	✓	✓	✓		✓					✓	✓
8	✓	✓	✓				✓				✓
9	✓	✓	✓				✓				✓
10		✓	✓	✓				✓			
11		✓	✓	✓				✓			
12		✓	✓	✓				✓			
13		✓	✓	✓				✓			
14		✓	✓	✓				✓			
15		✓	✓	✓				✓			
16		✓	✓	✓				✓			
17		✓	✓	✓				✓			
18		✓	✓	✓			✓				
19		✓	✓	✓			✓				
20		✓	✓	✓			✓				
21		✓	✓	✓			✓				
22		✓	✓	✓			✓		✓	✓	
23		✓	✓	✓			✓		✓	✓	
24				✓		✓				✓	
25							✓				
26							✓				
27		✓	✓	✓				✓			
28		✓	✓	✓				✓			
29		✓	✓	✓				✓			
30		✓	✓	✓				✓			
31		✓	✓	✓			✓				
32		✓	✓	✓			✓				
33		✓	✓	✓			✓			✓	
34				✓		✓				✓	
35							✓				

Table 6.5: OASIS3 transformations

6.1.2 Computing environment

All of the experiments have been made on two different architectures available at the CMCC Supercomputing Centre: a scalar cluster based on IBM Power6 processors and a vector cluster based on NEC-SX9 processors.

The IBM cluster, named Calypso, has 30 IBM p575 nodes, each of them equipped with 16 Power6 dual-core CPUs at 4.7GHz (8MB L2/DCM, 32MB L3/DCM). With Simultaneous Multi Threading (SMT) support enabled, each node hosts 64 virtual cores. The whole cluster provides a computational power of 18 TFLOPS. Each node has 128GB of shared memory (4GB per core), two local SAS disks of 146,8GB at 10k RPM and two Infiniband network cards each one with four 4X IB galaxy-2 and four Gigabit network adapters. Some nodes are used as GPFS and TSM servers and have also two fibre channel adapters at 4Gb/s FC and two fibre channel adapters at 8Gb/s for interconnecting to the storage system. Calypso has 2 storage racks, each one equipped with 280 disks of 750GB, providing a total capacity of 210TB of raw storage. These disks are working with GPFS. Calypso interconnects also a tape library with 1280 cartridges LTO4 at 800GB (1PB total capacity) and Tivoli TSM for handling Hierarchical Storage Management. The default compilers are IBM XL C/C++, and IBM XL FORTRAN. The default resource scheduler manager is LSF.

The NEC cluster, named Ulysses, has 7 nodes based on SX9 processors. Each node has 16 CPUs at 3.2GHz, 512GB of shared memory, a local SAS D3-10 disk of 3.4TB and uses IXS Super-Switch interconnection with a bandwidth of 32GB/s per node (16GB/s for each direction) to the high-speed interconnection and four 4Gb/s FC adapters to storage system. The whole cluster provides a computational power of 11.2 TFLOPS. Ulysses has 3 storage racks with three SAS D3-10 disks at 9.2TB and three SAS D3-10 disks at 6.9TB for a total capacity of 48.3TB of raw storage. The GFS is used for handling the storage system. The default compilers are SX C/C++ and SX FORTRAN. The default resource scheduler manager is NQSII.

6.2 Portability on IBM Power6 architecture

The porting activity on the IBM cluster consists of the following three steps:

A1: compilation, configuration and execution of the component models as they are executed on the vector cluster, without any code optimization.

A2: analysis of bottlenecks and definition of component models to be improved in order to optimize the coupled model performance.

A3: optimization of coupled model as result of the previous activity, taking into consideration the target architecture and the availability of native libraries performing better w.r.t. those actually used.

The version of ECHAM5 included within the CMCC-MED coupled model is optimized for the NEC-SX9 cluster and it is characterized by several physical improvements introduced by CMCC-INGV. Moreover, a stand-alone version of the atmospheric model provided by IBM and optimized for scalar architecture Power6, is available. In order to maintain the optimizations of the stand-alone version, an integration of the physical changes within it has been started. To date, the ECHAM5 stand-alone version has been coupled in the CMCC-MED model and we are working on the integration of the physical improvements. The Mediterranean component NEMO at 1/16° has been also developed by CMCC (IOIPSL provides a version of NEMO from 1° to 1/12°).

The porting activity of the CMCC-MED model on IBM Power6 is currently at stage A2. During stage A1, we used only the compiler optimization flags for tuning the model on the scalar architecture. Moreover, to compile CMCC-MED at IBM Power6 cluster, some modifications on the code were needed.

From a preliminary profiling analysis of ECHAM5 and NEMO component models we deduce that NEMO performances are limited by the communication overhead when open boundaries are activated and ECHAM5 does not scale well since we are using a version deeply optimized for vector clusters.

6.3 Performance analysis

In the High-Performance Computing context, the performance evaluation of a parallel algorithm is made mainly considering the elapsed time running the parallel application with different number of cores and different problem sizes (for scaled speed-up). Typically, parallel applications embed mechanisms for efficiently using the allocated resources, guarantying for example a good load balancing and reducing the parallel overhead. Unfortunately, this assumption is not true for coupled models. These models were born from the coupling of stand-alone climate models. The component models are developed independently from each other and they follow different development roadmaps. Moreover, they are characterized by different levels of parallelization, by different requirements in terms of workload and they have their own scalability curve. Considering a coupled model as a single parallel application, we can note the lacking of the implementation of a policy for balancing the computational load on the available resources.

During the scalability analysis of the CMCC-MED coupled model, we established a methodology for studying the scalability of parallel applications composed by independent parallel applications interacting each other with different communication mechanisms.

Coupled models are executed typically using a MPMD approach; established the total number of cores allocated for the whole coupled model, how we have to distribute them for each of the component models?

We have addressed this issue according to the following methodology:

1. Within the coupled model we analyzed the scalability curve of each component model and coupler.

2. We defined an analytic performance model at coarse grain level for the whole coupled model.

3. Using the experimental data given during the stage 1, and evaluating the model defined in stage 2, the best configurations have been extracted for different numbers of available cores.

4. We experimentally evaluated the behaviour of the coupled model considering only the best configuration for a given number of allocated cores.

In the next paragraphs, we will show the result from each stage of the proposed methodology on the two different architectures available at the CMCC.

6.3.1 Stage 1: scalability evaluation of the component models

The test aims at both studying the speed-up of each component model and finding the optimal run configuration in order to efficiently exploit the computing resources.

Several experiments have been performed to evaluate the scalability of each single component model. For each of these, we report the elapsed time for executing one-day simulation (it does not include the I/O time for writing the restart files).

Each model has been separately evaluated but within the coupled model. We used the PRISM libraries for instrumenting the code and for extracting the elapsed time of the single component models. We have considered the time elapsing between a prism_get and a prism_put as the time spent by the model for simulating all the time steps included in a coupling period. The coupling time has been evaluated considering the elapsed time between a clim_Import and a clim_export.

The components we have taken into account are ECHAM5, NEMO and the OASIS3 coupler that are the most computational intensive components. The OPA8.2 model is run in configuration ORCA2 using the sequential version. It is not been analyzed since it did not represent a bottleneck for any of the configurations taken into consideration.

All of the experiments have been performed using only MPI1 approach. Even if NEMO and ECHAM5 models support a hybrid parallelization based on OpenMP/MPI, in our experiments the number of threads for process has been set to 1.

In the following, we describe the analysis of scalability for each component model.

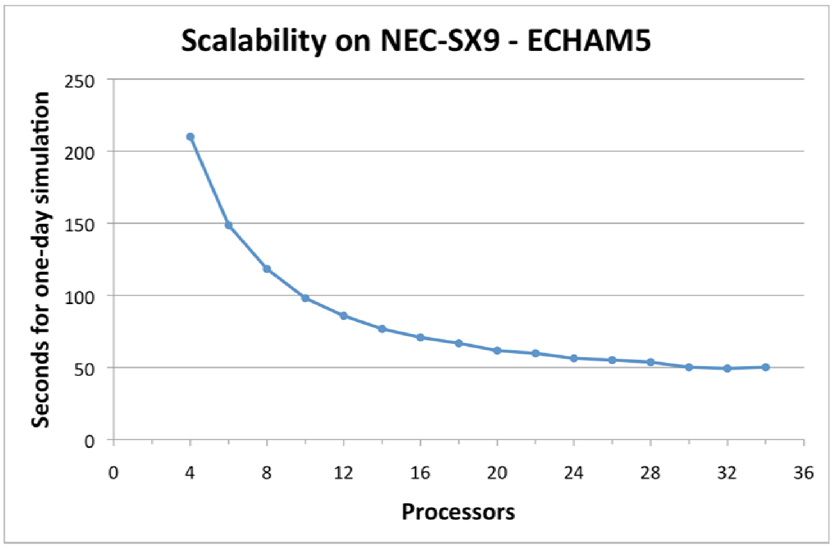
	Compilation flags	Linked libraries
NEC-SX9	-pi exp=cuentr,cuadjtq,cubasmc expin=cuentr.f90,cuadjtq.f90,cubasmc.f90 -Wf,-init heap=zero stack=zero -Popenmp -Chopt -sx9	libsupport.a libblas.a liblapack.a libpsmile.MPI1.a libmpp_io.a libnetcdf.a
IBM Power6	-qsmp=omp -q64 -O5 -qstrict -qarch=pwr6 -qtune=pwr6 -qcache=auto -qfixed -qsuppress=1518-061:1518-128 -qMAXMEM=-1 -Q -qsuffix=cpp=f90 -qzerosize -qessl -qnosave -qalias=nostd	libsupport.a libessl.a libmassv.a liblapack3264.a libxlf90_r.a libpsmile.a libmpp_io.a libnetcdf.a

Table 6.6: Compilation flags for ECHAM5

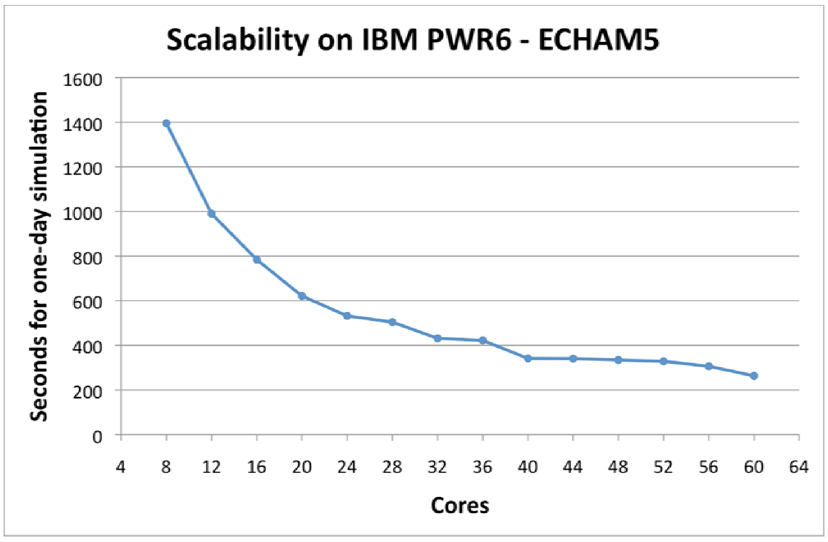
6.3.1.1 ECHAM5 component

The ECHAM5 component has been compiled on NEC-SX9 and IBM Power6 using the compilation flags in Table 6.6.

On NEC-SX9, ECHAM5 scales up to 28 processors (Figure 6.1a). On IBM Power6 some problems occurred when using block domain decomposition. Using only 1D decomposition, with a resolution of T159 (corresponding to 240x480 grid points), the maximum number of processes is 60. Hence, the scalability of ECHAM5 on IBM Power6 has been evaluated up to 60 cores (Figure 6.1b).



(a)



(b)

Figure 6.1: Scalability of ECHAM5 component model on NEC-SX9 (a) and on IBM Power6 (b) for one-day simulation

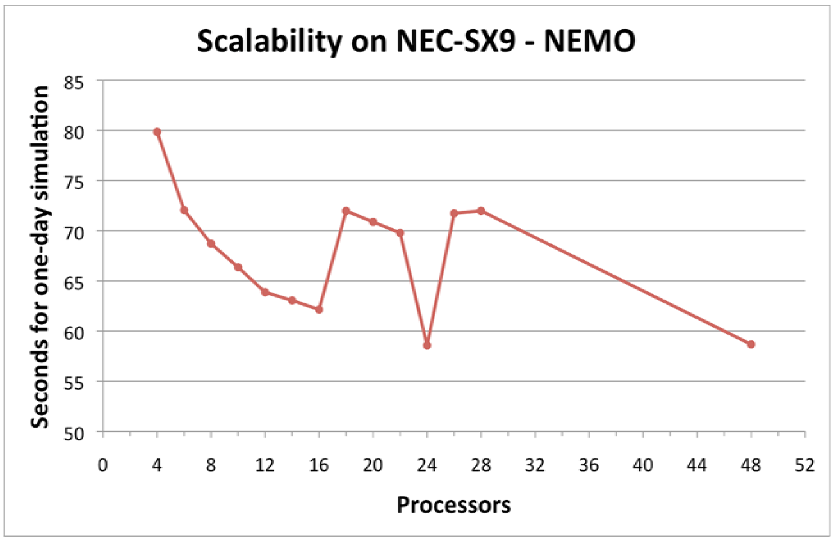
6.3.1.2 NEMO component

The NEMO component model has been compiled on NEC-SX9 and IBM Power6 using the compilation flags in Table 6.7.

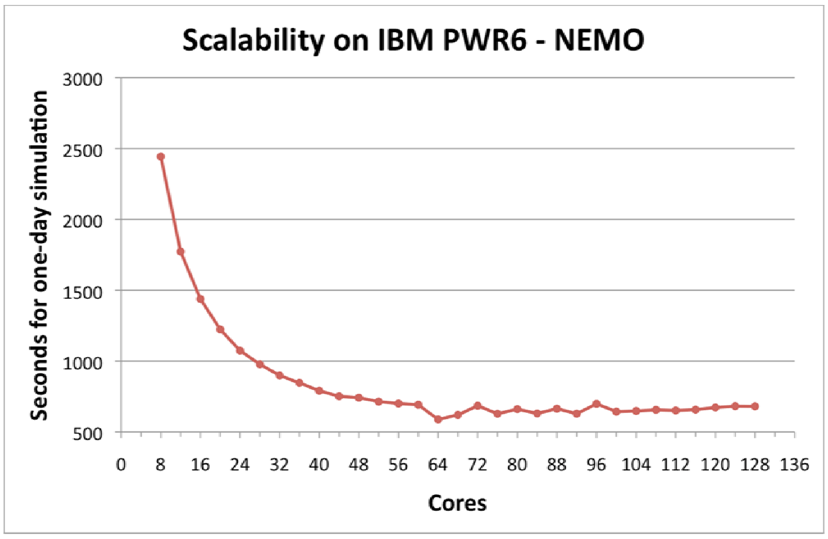
	Compilation flags	Linked libraries
NEC-SX9	-size_t64 -dw -Wf\"-A dbl4\" -sx9 -pi auto -P stack -C vopt -Wf"-init stack=nan" -Wl"-f nan" -Wf"-P nh" -Wf,-pvctl noassume loopcnt=10000	liboasis3.MPI1.a libpsmile.MPI1.a libmpp_io.a libclim.MPI1.a libnetcdf.a
IBM Power6	-q64 -c -qfixed -qrealsize=8 -qstrict -qarch=pwr6 -qtune=pwr6 -qcache=auto -O5	liboasis3.MPI1.a libpsmile.MPI1.a libmpp_io.a libclim.MPI1.a libnetcdf.a

Table 6.7: Compilation flags for NEMO

On the vector cluster, NEMO model presents a not regular trend in the scalability (Figure 6.2a). The scalability is limited to 24 vector processors. On IBM Power6, the model scales better up to 64 scalar cores (Figure 6.2b). It is worth noting here that the elapsed time on both machines differs of one order of magnitude.



(a)



(b)

Figure 6.2: Scalability of NEMO component model on NEC-SX9 (a) and on IBM Power6 (b) for one-day simulation

6.3.1.3 OASIS3 coupler

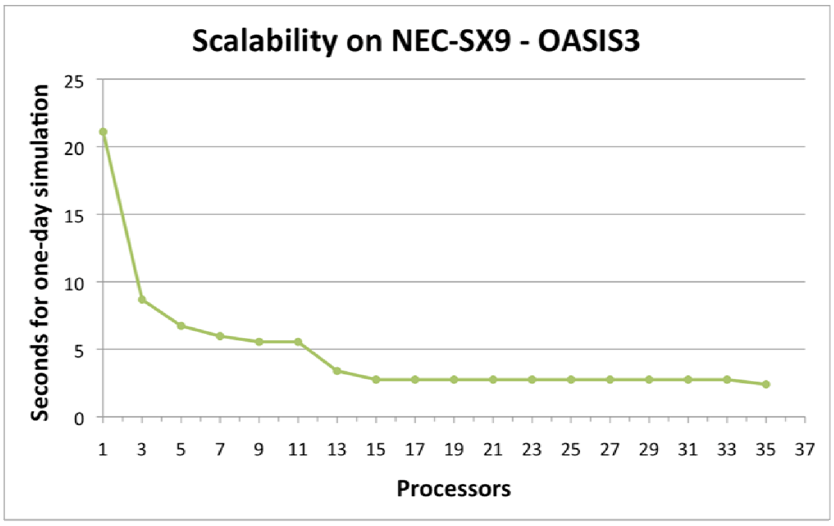
The OASIS3 coupler has been compiled using the compilation flags in Table 6.8.

	Compilation flags	Linked libraries
NEC-SX9	-Pstack -pi auto nest=3 line=10000 exp=iminim,rmaxim,rminim,grid_search_bilin expin=src/,libsrc/scrip -Ep -sx9 -Wf,"-P nh" -Wf,"-pvctl noassume loopcnt=5000000 vworksz=100M" -Wf,"-A idbl4" -Wf,"-msg o" -Wf,"-pvctl fullmsg" -Wf,"-L fmtlist transform map summary noinclist" -Chopt -Wf,"-ptr byte"	libpsmile.a libanaisg.a libanaism.a libfscint.a libscrip.a libclim.MPI1.a libmpp_io.a

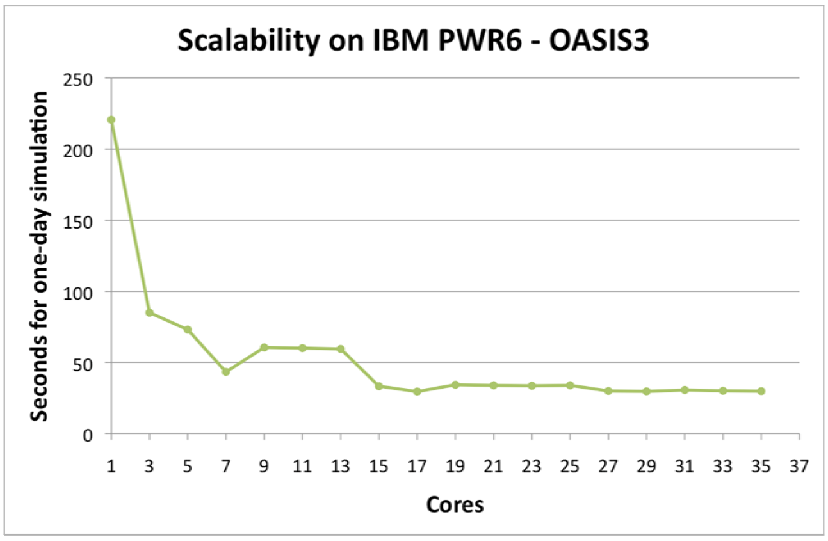
	Compilation flags	Linked libraries
IBM Power6	-qrealsize=8 -q64 -qalias=nostd -O0 -qsuffix=cpp=F90	libpsmile.a libanaisg.a libanaism.a libfscint.a libscrip.a libclim.MPI1.a libmpp_io.a

Table 6.8: Compilation flags for OASIS3

The parallel approach used in the CMCC version of OASIS3 follows an embarrassing parallel algorithm. The scalability on both architectures shows that the communication overhead is negligible. The not linear trend of the scalability, corresponding to some number of processes, is due to the not balanced workload on different fields (Figure 6.3). The parallel algorithm limits the scalability to the number of exchanged fields (in our case it is equal to 35).



(a)



(b)

Figure 6.3: Scalability of OASIS3 coupler on NEC-SX9 (a) and on IBM Power6 (b) for one-day simulation

6.3.2 Stage 2: definition of the performance model

For the definition of the analytic performance model, we can consider that all the model components are executed in parallel between two coupling steps. At each coupling step, each model sends their fields to the coupler and waits for receiving from the coupler the fields coming from the other models. The coupler, only after receiving all of the fields to be exchanged, performs their transformations, sends the changed fields to the models and waits for receiving fields at the next coupling step. During the coupling period all the models are synchronized and waits for coupler ending transformations (coupler transformations and models elaboration could be overlapped using the OASIS3 utility named SEQ. During the analysis of the CMCC-MED performance, this utility has not been used). The analytic model defines the execution time for a single coupling step as the maximum computing time spent by the component models plus the coupling time (the last time step of NEMO, performed between a coupling step and the next one, is overlapped with the coupling activity for modellers chosen):

$$T = \max\left\{t_N(p_N) * (n_{s_N} - 1), t_E(p_E) * n_{s_E}\right\} + \max\left\{t_O(p_O), t_N(p_N)\right\} \quad (1)$$

where

t_N is the computing time for executing a time step of NEMO component model

t_E is the computing time for executing a time step of ECHAM5 component model

n_{SN} is the number of time step of NEMO component model

n_{SE} is the number of time step of ECHAM5 component model

t_O is the computing time for one coupling step

p_N is the number of processes allocated for NEMO component model

p_E is the number of processes allocated for ECHAM5 component model

p_O is the number of processes allocated for OASIS3 coupler

In order to find the best configuration we have to find the p_E, p_N and p_O values minimizing the execution time T. The constraint imposes that the total number of processors must be equal to the allocated processors (K).

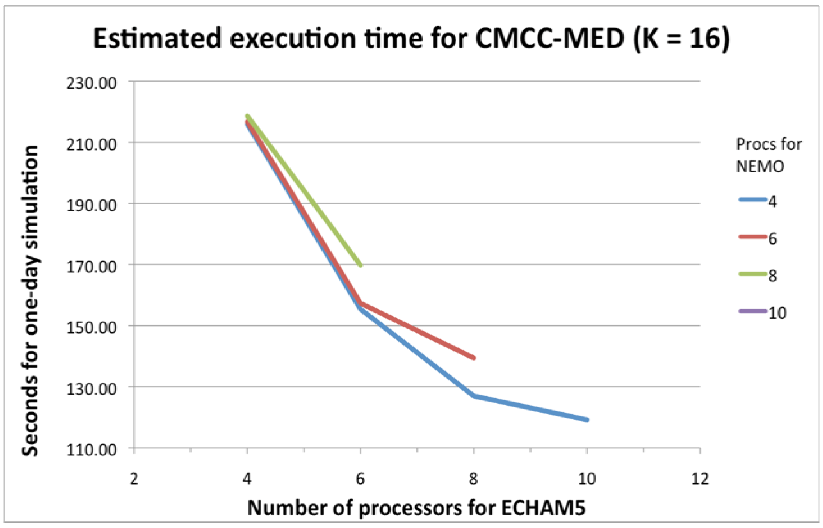
$$p_E + p_N + p_O + 1 = K \quad (2)$$

6.3.3 Stage 3: definition of the best configurations

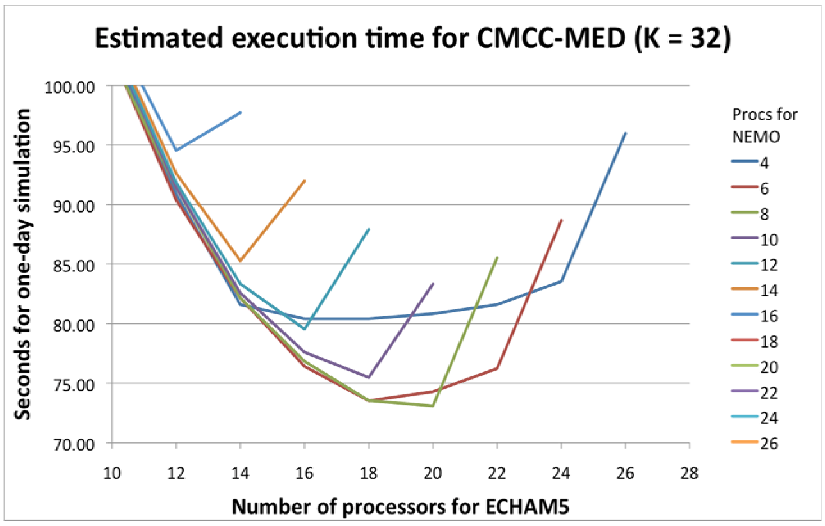
During this stage we used the experimental results obtained in the stage 1 for evaluating the performance model of the whole coupled model. The charts have been produced setting K to the total number of CPUs/cores to be used and evaluating the performance model for all of the permutations of p_E, p_N, and p_O satisfying equation 2.

Taking into account the time step intervals and the coupling period (reported on Table 6.3 and Table 6.4) we have $n_{SN} = 16$ and $n_{SE} = 40$

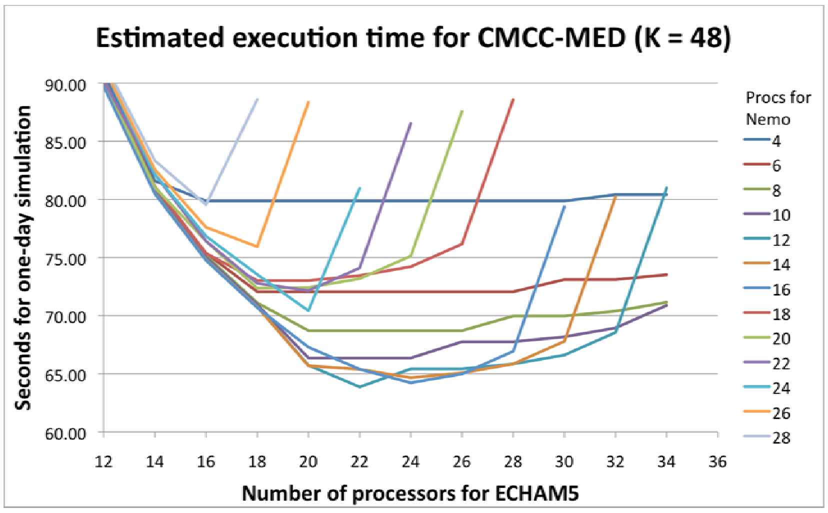
6.3.3.1 Best configurations on NEC-SX9



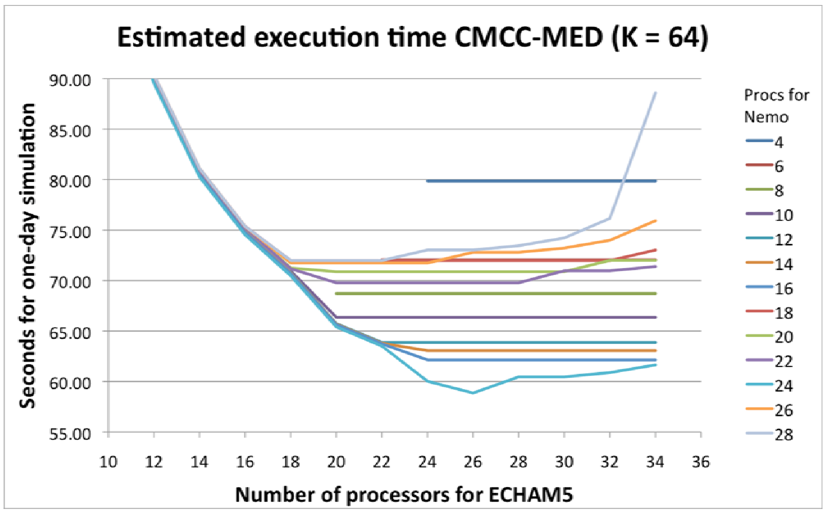
(a)



(b)



(c)



(d)

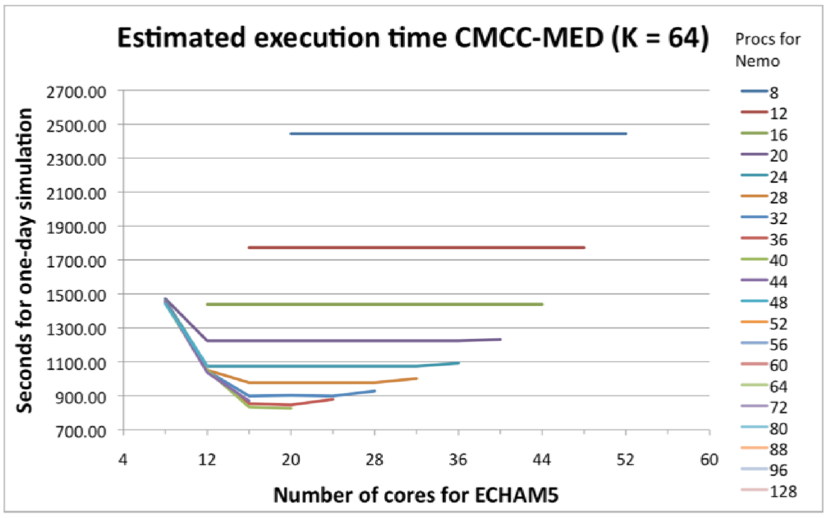
Figure 6.4: Estimated execution time of the CMCC-MED on NEC-SX9 for K=16 (a), K=32 (b), K=48 (c) and K=64 (d)

The best configurations are represented in the Table 6.9.

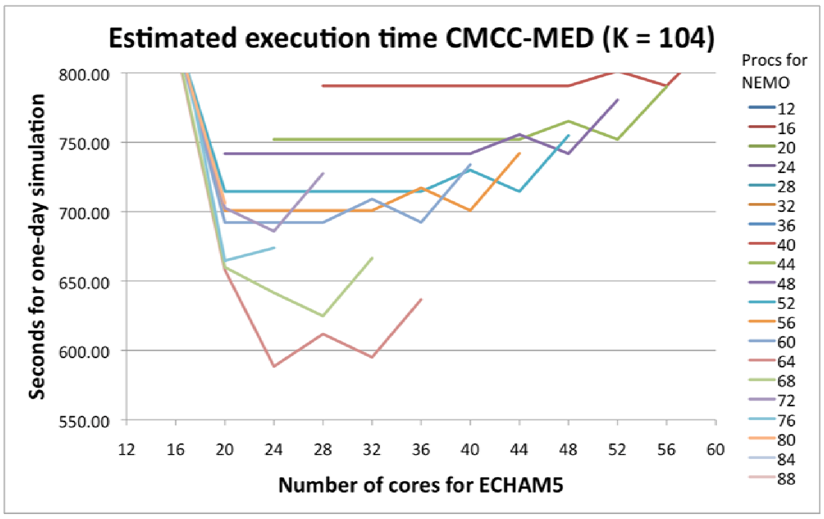
K	p_E	p_N	p_O
16	10	4	1
32	20	8	3
48	22	12	13
64	26	24	13

Table 6.9: Best configurations on NEC-SX9

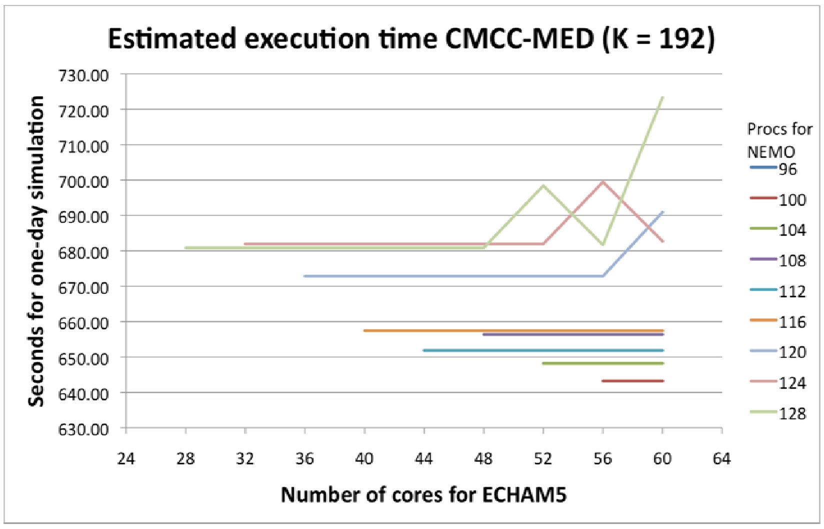
6.3.3.2 Best configurations on IBM Power6



(a)



(b)



(c)

Figure 6.5: Estimated execution time of the CMCC-MED on IBM Power6 for H=64 (a), K=104 (b) and for K=192 (c)

K	p_E	p_N	p_O
64	20	40	3
104	24	64	15
192	60	100	31

Table 6.10: Best configurations on IBM Power6

6.3.4 Stage 4: scalability trend of CMCC-MED model

Considering the trend of the performance model, evaluated on the NEC-SX9 (Figure 6.6), we can make the following considerations:

1. The execution time could decrease with a greater number of processors. The actually availability of processors on NEC-SX9 limited the analysis of scalability to 4 nodes.
2. Established the number of nodes to be used, the best performance have been obtained when the allocated nodes are fully used.

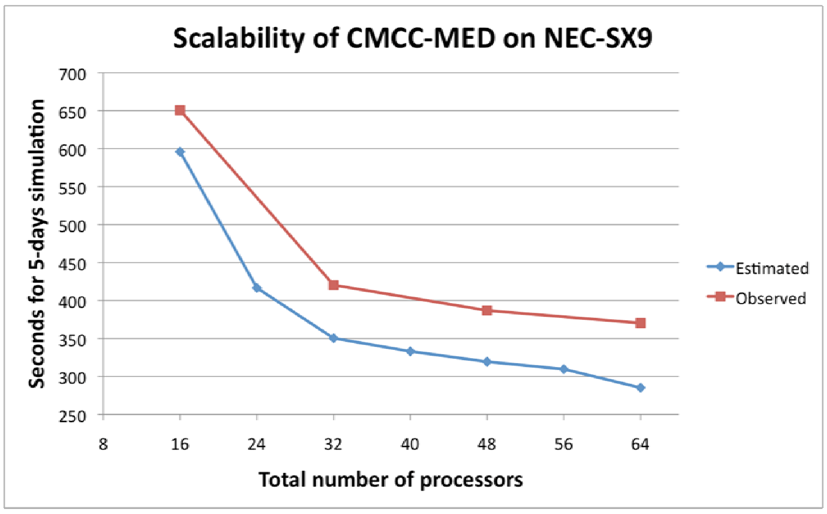


Figure 6.6: Scalability of CMCC-MED on NEC-SX9

On IBM Power6 cluster (Figure 6.7) we can make the following considerations:

1. The limit of the scalability has been reached at 104 cores. Even if the cluster provides a greater number of nodes, our analysis stopped at 3 nodes since we have reached the minimum elapsed time.
2. Established the number of nodes to be used, the best performance have been obtained not always when the allocated nodes are fully used, i.e. with 3 nodes it is not necessary to use all 192 cores, being 104 enough (the elapsed time on 104 and 192 is the same).

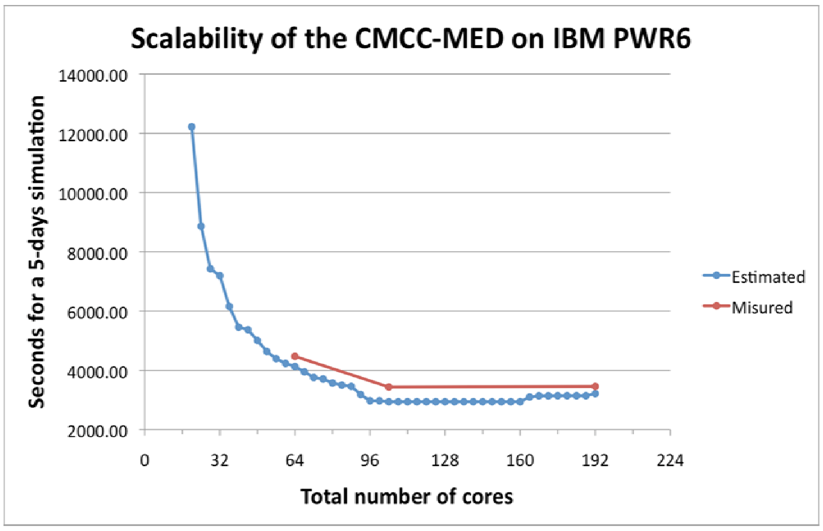


Figure 6.7: Scalability of CMCC-MED on IBM Power6

The scalability of the coupled model has been experimentally evaluated varying the number of nodes and using the best configurations suggested by the performance model. The experiments take into account a 5 days simulation and include the I/O time for writing the restart files. The results confirmed the likelihood of the performance model with the real computational behaviour of the coupled model, as shown on Figure 6.6 and Figure 6.7.

PART III: Concluding remarks

1. Summary of the base-case results

The deliverable reports results obtained from four couple models (CMCC-MED, ARPEGE-NEMO, IPSL-ESM and EC-Earth) and four stand-alone models (NEMO, ARPEGE, LMDZOR and ECHAM5) tested on a number of HPC infrastructures within the IS-ENES network (e.g., Ekman, Kappa, MareNostrum, CINES, IDRIS, Calypso, Ulysses). The collaborative effort among application owners and computer specialists led to the identification of numerous strengths and limitations of these ESMs and models. The main findings are summarized below:

- NEMO stand-alone model:
  - The ORCA1 benchmark scales well to around 16 nodes (128 cores) on Ekman (Quad-Core AMD Opteron 2374 HE @ 2.19 GHz; 8 MPI processes / node) and Kappa (Quad core Intel(R) Xeon(R) CPU E5520 @ 2.27GHz, code named Nehalem; 8 MPI processes / node) systems. The scaling is gradually lost due to the large amount of the time is spent in MPI calls.
  - The choice of the MPI library has some impact on run time of NEMO code. Open MPI runs are consistently better than Scali MPI. The different behaviour may be due to some MPI internal algorithm differences.
- LMDZOR atmosphere stand alone model:
  - The stand alone atmospheric model was tested on MareNostrum (PowerPC 2,3 GHz, 4 CPU / node, 8 Gb / node), Vargas Power6 (4,7 GHz, 32 CPU / node, 128-256 Gb / node) and Mercure SX8R (vector processors, 8 CPU / node, 64 Gb / node). The speedup is lower for PowerPC than for Power6. Similar behaviour can be obtained on Power6 compared with vector processors by increasing the number of cpus used.
- ARPEGE-NEMO coupled model and OASIS3:
  - Several configurations of ARPEGE-NEMO have been set up in order to check OASIS3 capability at high end resolutions. Tests were performed on Météo-France supercomputer (6+7 vector nodes of 16 SX9 processors each and 1 Tb memory per node).
  - An internal counter allows to finely determine the number of processors which minimizes the difference between oceanic and atmospheric coupling step durations.
  - Two optimizations on OASIS configuration (sequential mode + pseudo parallelism) speed up the coupled standard runs between 15 and 25%. A higher resolution is tested to analyse OASIS3 performance. Both optimizations contribute to reduce the total simulation time from 8 to 4 hours. Those experiments prove the OASIS3 capability to drive high end resolution coupled simulation on vector machines.
- ARPEGE-NEMO coupled model:
  - The coupled model was ported on SGI Altix ICE (CINES), IBM Blue Gene (IDRIS) and Marenostrum IBM JS21 (BSC) to analyse if the model can intensively be used on scalar machines. Major problems were encountered on Blue Gene architecture. OASIS3 cannot be used without major changes on such architecture. No relevant problems were detected on the other platforms.
- IPSL-ESM coupled model:

    - The porting of IPSL-ESM to MareNostrum (IBM Power PC 970MP @ 2.3 GHz; 4 MPI processes / node; 8 Gb / node) supercomputer highlights some limits of the model performance in such type of architectures: low memory available per node, scalar processors with low frequency calculation, moderate MPI scalability, no hybrid MPI-OpenMP runs allowed.
    - A higher resolution configuration was ported on CINES-Jade (2 Intel Quad-Core E5472 processors; 30 Gb / node). The hybrid parallelisation MPI-OpenMP allows running the coupled model on around 2200 cores.
- EC-Earth coupled model:
    - The EC-Earth ESM was successfully ported to Marenostrum (PowerPC 2,3 GHz, 4 CPU / node, 8 Gb / node) supercomputer. The atmospheric component of the system scales well to around 64 cpus. The current configuration of the system did not allow to run NEMO code with more than 16 cpus.
    - The heterogeneous parallelisation MPI and MPI+OpenMP is not currently allowed on the Marenostrum system. This is an important limitation on this architecture as it prevents taking advantage of the speedup of NEMO when run using MPI+OpenMP.
- CMCC-MED coupled model:
    - The coupled model was ported and tested on Calypso (IBM Power6 dual-core @ 4,7 GHz) scalar machine and Ulysses (NEC SX9 processors, 16 cpu / node @ 3.2 GHz, 512 Gb of shared memory) vector machine.
    - The coupled model scales well to around 32 cpus on Ulysses machine and 104 cpus on Calypso. On cluster machines, the best performance is not always obtained when the allocated nodes are fully utilised (maximum performance may be on fewer cores than are allocated on a node-basis).

2. Recommendations for future work

Several strategies are planned in LIU, IPSL, BSC, CERFACS and CMCC for future work.

At **LIU**, they plan continuing to profile NEMO, including MPI calls along with other functions and subroutines using different profiling tools. In addition, they will put some light weight timers in the code to analyze its behaviour and learn in detail about potential load imbalances and other bottlenecks. They plan to analyze scaling of main time consuming routines and figure out which routines are loosing scaling as the number of ranks increases. Finally, they will investigate whether it will be possible to reduce the MPI_Recv time.

In the next months, **IPSL** will continue working on "Grand challenge" project at CINES centre, in order to run 20 years simulation at high resolution (see part 1.2 of IPSL report). They plan to continue to test their model on SMP and MPP architectures to evaluate the work to be done to obtain good performances on such a machines, especially concerning following points:

- Hybrid MPI-OpenMP parallelisation on MPMD code4: it is not easy to specify a number of OpenMP threads and MPI process on most of architectures. For example, they would like to launch easily the following configuration (especially because the parallelization hybrid MPI-OpenMP seems to be the best way to use up the resources available on MareNostrum-like architecture, see IPSL report).
  - Atmospheric model on 2048 cores by using hybrid MPI-OpenMP parallelization : 256 MPI process x 8 OpenMP threads
  - Oceanic model on 120 MPI process
  - Oasis3 parallel version "field per field" on 24 MPI process (each MPI process treats one field).
- I/O problems: the introduction of different levels of parallelization into models has resulted in a significant decrease in computing time. As a consequence, time spent on I/O becomes very significant proportion of total computing time (15-20%). Simulation performance is highly impacted by time spent in the writing model output. They have planned to work on this problem both at IPSL and through IS-ENES project WP7/JRA1.

At **BSC**, they plan to continue the trace analysis of the coupled EC-Earth run with Paraver. They will focus on the improvement of the execution of NEMO code in MareNostrum and, if possible, test the use of a hybrid parallelization MPI-OpenMP to reduce the communication time detected within NEMO. Defining a higher resolution configuration within EC-Earth community will allow them to analyze the performance of the model beyond thousands of CPUs. For that, the implementation of a parallel version of the OASIS coupler will be critical.

With the experience with EC-Earth model and an optimized implementation of NEMO and OASIS on MareNostrum, BSC will collaborate with other modelling groups (e.g., IPSL, CERFACS) to test their models on higher resolutions.

In **CERFACS**, one of the major challenges for the upcoming months is the set up of an OASIS4 configuration to test scalability of their different model components on a coupled configuration. The aim of this task is to estimate OASIS4 performances in an actual MPP configuration (with high resolution components) and validate, with a scientific study, the good operability of a high resolution coupled model (global CGCM) on a scalar platform.

The work began with the implementation of ECHAM-NEMO coupled system on both IDRIS-CNRS and DKRZ IBM Power6 (WP4, OASIS user support) and will follow on with the new implementation of ARPEGE / NEMO-Mixed-Layer CGCM (also called ARPEGE-PanOPAe) on CINES SGI Altix ICE (see IPSL prospective) and IDRIS IBM Power6.

A similar work is planned with EC-Earth system (high resolution) during WP4 user support effort at SMHI. Strong interactions on this topic are expected with both LIU and BSC partners.

At **CMCC**, they plan improving scalability of CMCC-MED coupled model on IBM Power6. Performance decreases for both ECHAM5 and NEMO-Mediterranean component models when the number of processes increases; this influences the behaviour of the coupled model. In particular, ECHAM5 shows some problems when block domain decomposition is used: using only 1D decomposition, with a resolution of T159, the maximum number of processes is 60. Hence, they plan to solve decomposition problem and to evaluate scalability of ECHAM5 on IBM Power6 beyond 60 processes. The NEMO-Mediterranean component scales better up to 64 scalar processes due to load imbalance and communication increasing for a higher number of processes. They plan to investigate bottlenecks to performance improving and to perform a full scalability analysis running NEMO on more then 60 processes.

References

Hazeleger, W. et al., 2009. EC-Earth: A Seamless Earth System Prediction Approach in Action, accepted, Bull. Amer. Meteor. Soc.

Martijn Brandt, EC-EARTH- the European Community Earth system model, March 2010 [http://ecearth.knmi.nl/EC-Earth_model_documentation.pdf]

Stefanescu S., Standalone environment for compiling and running the EC-EARTH system, Technical Note, April 2008 [http://ecearth.knmi.nl/ecearth2.pdf].