IS-ENES - WP8

D8.4 - Final Report on Portability and Performance in IS-ENES ESMs

Abstract

Grant Agreement Number:	228203	Proposal Number:	FP7-INFRA-2008-1.1.2.21
Project Acronym:	IS-ENES		
Project Co-ordinator:	Dr Sylvie JOUSSAUME		

Document Title:		Final Report on Portability and performance in IS-ENES ESMs		Deliverable:	D8.4
Document Id N°:		Version:	2.0	Date:	16/01/2012
Status:	Final				

Filename:	ISENES_D8.4_ESM_Portability_and_Performance_Final_Report.doc

Project Classification:	Public

Authors	
O. Jorba, K. Serradell, L. Telloli	BSC (16)
O. R. Darbyshire, A.M Clayton	METOFFICE (10)
G. D. Riley -	UNIMAN (6)
C. Basu	LIU (15)
A.Caubel, Y.Meurdesoif, O.Marti	CNRS-IPSL (1)
E. Maisonnave, P-A, Bretonniere. C. Cassou, S. Valcke, T. Craig	CERFACS (3)
J. Silen	FMI (5)
J. Lento	CSC, subcontract to FMI (5)
I. Epicoco	University of Salento
S. Mocavero	CMCC (9)
G. Aloisio	CMCC (9)- University of Salento

REVISION TABLE

Version	Date	Comments	Authors, contributors, reviewers
0.1	16/1/2012	First draft: outline and inclusion of contributions	G.D. Riley and Contributors listed above
0.2	14/2/2012	Including reviewer responses	G.D. Riley and Contributors
1.0	30/3/2012	Including final reponses to reviewer comments	G.D. Riley, Contributors and S. Valcke, G. Aloisio.
1.1	21/1/2013	Addition of CMCC and FMI/CSC contributions	G.D. Riley
2.0	27/2/2013	Addition of contributions from LIU, SMHI, METOFFICE and responses to reviewer comments	G.D. Riley and contributors.

Table of Contents

PART I: Overview

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, including PRACE machines, 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 in terms of both porting and performance. This report is an interim report building on results presented in JRA2 deliverable, D8.2 "Evaluation suite and base-case results". A final report is due at the end of the project.

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 interim results obtained since the first performance report, D8.2, 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.

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 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 are, and will continue to be, able to take advantage of the PRACE initiative.

Purpose

The purpose of the present report is to summarize recent work undertaken to understand the performance of the ESMs of the IS-ENES 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. This report is a follow-on report to "**D8.2 Report on the Description of the Evaluation Suite and Base-case Results"** and builds on lessons learnt in the work described in D8.2. 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,
- attempts to improve both the portability of models, by documenting experiences of porting, where appropriate, and the performance of models on specific architectures.

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

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:

- IPSL, Performance Analysis and Portability of IPSL ESM model. A.Caubel.
- CERFACS: ARPEGE-NEMIX: a simplified high resolution CGCM for validation purpose on PRACE tier-0/tier-1 machines. E. Maisonnave, P.-A. Bretonnière, C. Cassou.
- CERFACS: ARPEGE-NEMIX porting, optimization and performance tests on PRACE tier0-tier1 machines. E. Maisonnave, S. Valcke, T. Craig, P.-A. Bretonnière.
- BSC: EC-Earth modelling system on MareNostrum Supercomputer: porting and performance experience.
- LIU/SMHI: High resolution EC Earth porting, benchmarking on CURIE. C. Basu. Including a section on ec-conf from SHMI.
- CMCC: NEMO Porting, benchmarking and optimization on Marenostrum. I.Epicoco, S.Mocavero, G.Aloisio.
- FMI: Porting and performance analysis of Cosmos-Millennium on Cray-XT5 at FMI. J. Silen.
- Porting and performance analysis of Echam6 on Cray XT4 at CSS (subcontractor to FMI). J. Lento.
- METOFFICE: UM Scaling on HeCToR. O. R. Darbyshire plus a report on technical issues with the UPSCALE PRACE project by S. Mullerworth.
- METOFFICE: Reducing the Sensitivity of the Met Office Unified Model to Rounding Errors. O. R. Darbyshire, A. M. Clayton.

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, and especially for the developing relationship with PRACE and use of PRACE tier-0 and tier-1 machines.

PART II: Porting and Performance reports

Performance Analysis and Portability of IPSL ESM model

Contributors: A.Caubel, Y.Meurdesoif, O.Marti[1]
[1] Institut Pierre Simon Laplace, France (IPSL)

Introduction

As presented in the IS-ENES D8.2 - Report on the Description of the Evaluation Suite and Base-case Results, the IPSL climate model was ported on the super computer Jade at CINES and tested at high resolution: atmosphere at 768x767x39 resolution, ocean at 1442x1021x75 resolution, using 2191 cores. An important effort to optimize the code and to reduce its memory footprint was undertaken at this time.

Through the PRACE preparatory access project "COUAC", this prototype version of the IPSL climate model was also ported on Curie (a PRACE machine) which allowed us to compare the performances on Jade and Curie.

In addition, some work on the environment of IPSL ESM standard version (CMIP5 version) was done in order to run production simulations of the model on new machines.

Performances of IPSL ESM model on Curie (PRACE machine) and Jade (CINES centre)

Three different configurations of IPSL model were tested:

IPSL standard low resolution coupled model (CMIP5 version) on Curie "Fat nodes"

- LMDZ Atmospheric model at 96x95x39 resolution (MPI parallelized version)

- NEMO Oceanic model 2° resolution i.e. 182x149x31 (MPI parallelized)

- Oasis3 sequential version: 1 coupling per day between Atmospheric model and Oceanic model.

Performance obtained on 32 CPUs (1 Oasis+ 5 NEMO +26 LMDZ MPI) are: Real time: 1380s/month

IPSL atmospheric high resolution model on Curie "Fat nodes" (and comparison with Jade)

In a typical climate model experiment, the CPU time used by LMDZ is 5 to 20 times larger than the CPU time used by other components. The number of cores used for each component is adequately chosen to allow a proper load-balancing. For this reason, we spent most of our time on LMDZ alone, to port the model on Curie, test the performance, and make some improvements.
We made the adaptation of LMDZ for the new Curie computer and run high-resolution test cases. For this, we used the same resolution 768x767x39 that has been run for "CINES grand challenge" on Jade computer (Intel Nehalem EP, 2.93 GHz) in 2010, so we can compare performance and scalability.

The test case was a global run on 768x767 latitude-longitude points with 39 vertical layers. We run 960 time step iterations by test-case, so about 4.8 hours of simulated times. We ran up to 256 MPI processes with 4 or 8 threads by process, so up to 2048 cores.

Results for 8 threads by process on Curie computer:

Core numbers	Elapsed	Speed-up
128	670	128
256	328	261
512	186	461
1024	129	665
2048	98	875

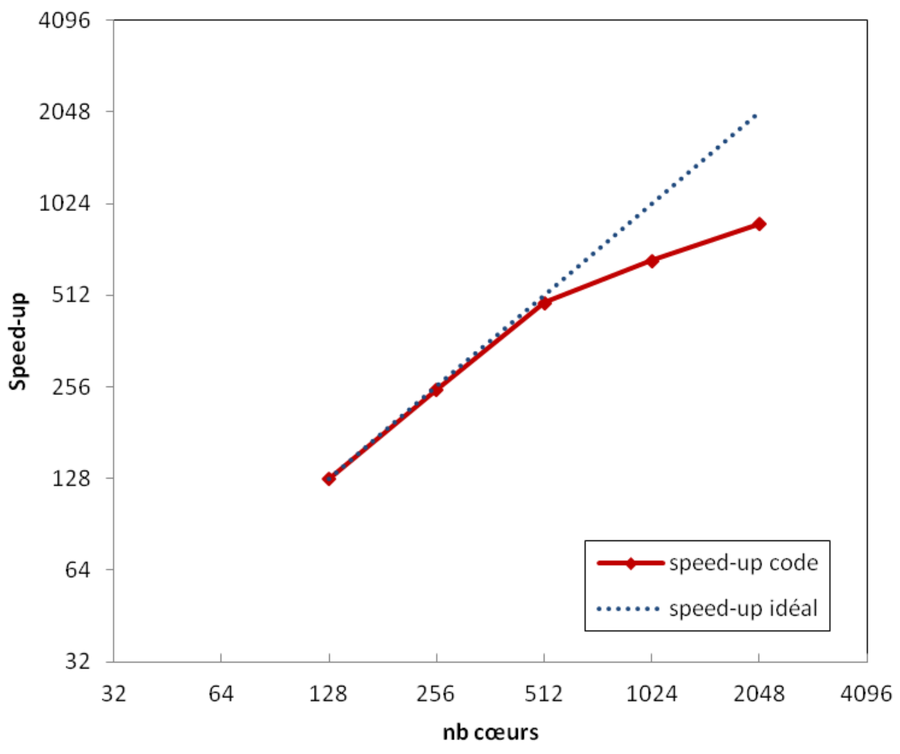


Figure 1: Speed-up for 8 threads by process on Curie. Red : code speed-up. Blue dotted : ideal speed-up.

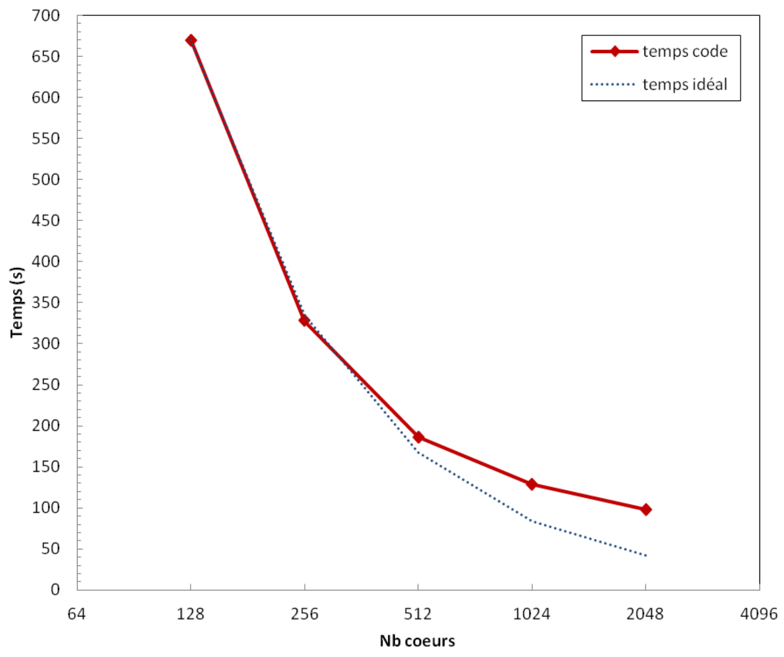


Figure 2: Elapsed time for 8 threads by process on Curie. Red : code speed-up. Blue dotted : ideal speed-up.

Results for 4 threads by process on Curie computer:

Core numbers	Elapsed	Speed-up
128	632	128
256	303	266
512	174	465
1024	117	691

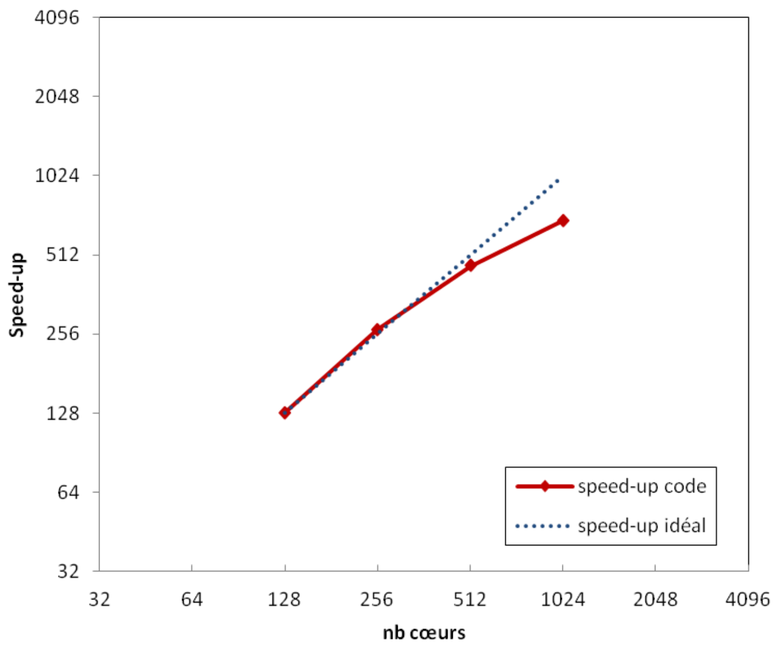


Figure 3: Speed-up for 4 threads by process on Curie. Red : code speed-up. Blue dotted :

ideal speed-up.

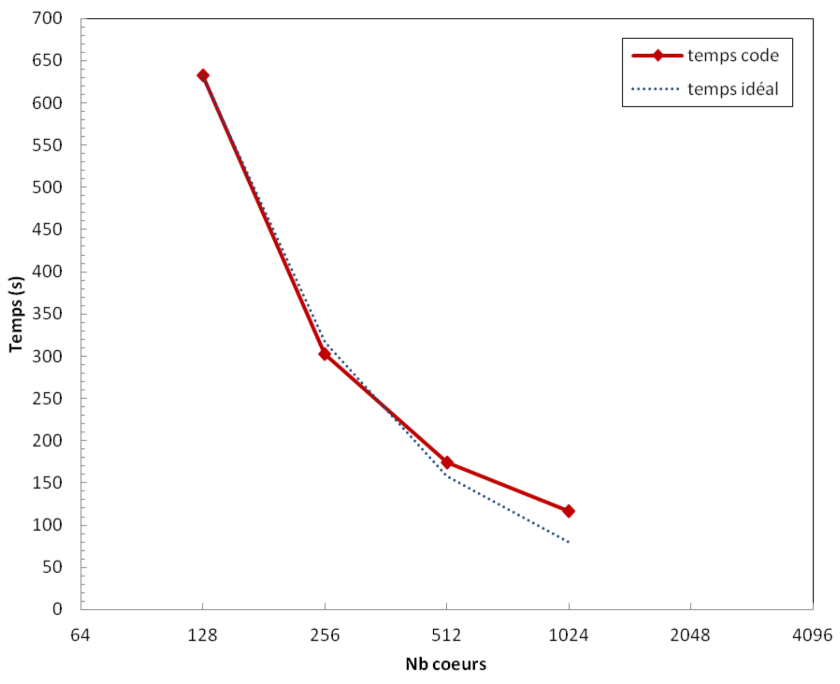


Figure 4 : elapsed time for 4 threads by process on Curie. Red : code speed-up. Blue dotted : ideal speed-up.

For comparison, results obtained on **Jade (CINES)**

Core numbers	**Elapsed**	**Speed-up**
64	924	64
128	440	134,4
256	220	267,36
512	122	484,7
1024	72	822
2048	49	1207

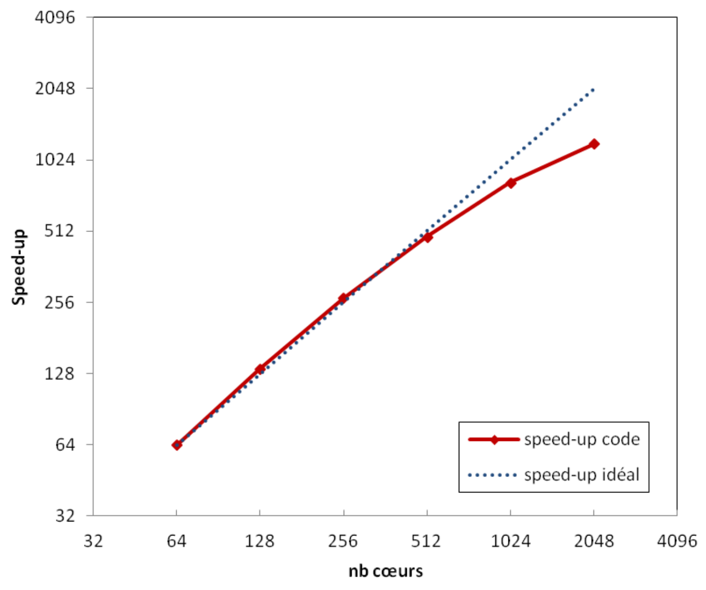


Figure 5 : Speed-up for 8 threads by process on Jade. Red : code speed-up. Blue dotted : ideal speed-up.

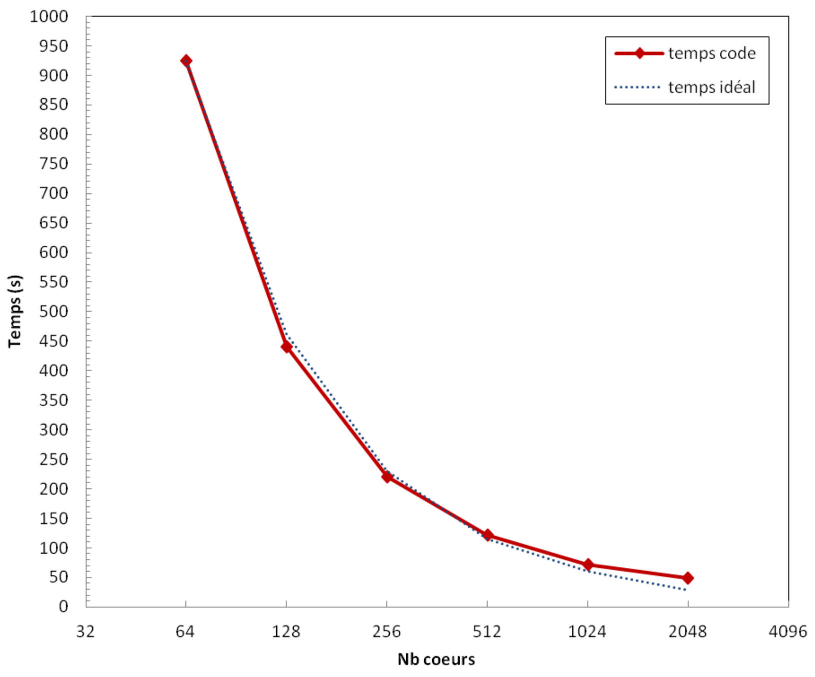


Figure 6 : Elapsed time for 8 threads by process on Jade. Red : code speed-up. Blue dotted : ideal speed-up.

It seems Jade results outperformed Curie results, in core-by-core comparison, for CPU time and for scalability. We can explain the difference on CPU time (~50% more on Curie) by the clock frequency difference (2.93 vs. 2.23, ~30%) and for a lesser memory bandwidth by core on Curie. For scalability, the network bandwidth by core is lesser on a Curie node than on a Jade node, so it may explain the difference. Other possibility is it may due to the quality of the network interconnect and/or quality of the MPI implementation. It is only supposition and we don't investigate anymore.The model will be soon ported on Curie "thin nodes", where we expect better performance. As we don't need large memory by core neither by node, and don't need large node, Curie "large nodes" doesn't seem to be the optimal machine for our case. But it has to be confirmed on Curie "thin nodes".

IPSL high resolution coupled model on Curie "Fat Nodes"

- LMDZ Atmospheric model 0.3°x0.3°(39 vertical levels) i.e. 768x768x39 (hybrid MPI-OpenMP parallelized)
- Oceanic model 0.25°x0.25°(75 vertical levels) i.e 1441x1021x75 on 120 MPI process
- Oasis3 parallel version "field per field" on 23 MPI process (each MPI process treats one field) 1 coupling per 2h between Atmospheric model and Oceanic model.

Performances obtained are :

- 23 OASIS + 120 NEMO + LMDZ 256MPIx4OMP = 1167 CPUs : Real time : 7,6h/month
- 23 OASIS + 120 NEMO + LMDZ 256MPIx8OMP = 2191 CPUs : Real time : 7h /month

Note the relatively small performance improvement in Real time when using 8 OpenMP threads rather than 4 (from 7,6h/month to 7h/month). Only preliminary tests were run to validate technical aspects of high resolution configuration, this means that an optimization phase is needed to improve performances, especially concerning heterogeneous configuration aspects in MPMD mode with the ocean model MPI parallelized and the atmosphere model mixed MPI-OpenMP parallelized.

Compilation and running environment of IPSL ESM model (production environment)

Work was done on the IPSL environment (compilation and execution) in order to increase the number of machines used to run our production simulations.

- Because of the end of vector machines as production machines, we have worked (and are still working) on running all of our production runs on IBM Power 6 (vargas - IDRIS centre) and Bull (titane - CCRT centre). This work consists both in :
  - ➢ quality checks : reproducibility, parallelization.
  - ➢ performance developments :
    - o use of hybrid parallelization MPI-OpenMP in production version of IPSL ESM model. The use of hybrid parallelization MPI-OpenMP on an heterogeneous configuration (MPMD model) needs both to modify our usual way to launch the model and to interact strongly with computing centres.
    - o analyze of load balancing between different component of IPSL ESM model (standard version=ATM_LMDZ_96_95_39xOCE_NEMO_ORCA2).

      Results obtained :
      - ✓ titane : IPSLCM5A (1day =coupling period) : lmdz :35s, nemo : 39s
      - ✓ vargas : IPSLCM5A (1day =coupling period) : lmdz :32s, nemo : 32s

- The IPSL compilation environment was installed on Curie. The installation of IPSL model running environment in order to run production simulations is planned.

Conclusions

The porting of IPSL -ESM model on new and different machines is very instructive both in terms of performances and usability. Besides, we have tested several configurations of the model at different resolutions: that helped highlight the work done in recent years both in terms of optimization and parallelization of the components (see the results in Section 0 obtained for high resolution IPSL model on Curie and Jade) and in terms of portability of production environment (compilation, execution) of the model (see the work reported in Section 0 to run the IPSL ESM standard version in production simulations on new machines Vargas, Titane and Curie). In order to use hybrid parallelization (MPI-OpenMP) on an heterogeneous configuration (MPMD model) we need to interact strongly with computing centres. In the future, the ideal situation would be that the computing centres actually communicate with each other.

ARPEGE-NEMIX: a simplified high resolution CGCM for validation purpose on PRACE tier-0/tier-1 machines

Contributors: E. Maisonnave[1], P.-A. Bretonnière[1], S. Valcke[1], T. Craig[1], C. Cassou[1]
[1] Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, France (CERFACS)

Introduction

An atmosphere-ocean coupled model is a complex, highly non-linear system, for which results are non reproducible from one machine to another. This forbids any bit-to-bit cross-computer validation comparing simulations with identical experimental conditions.
A conventional strategy to address this problem consists in comparing statistical results of simulation ensembles. Nevertheless, this solution requires a large amount of computing time (particularly at high resolution, which is mandatory on Petascale architectures) to reach the equilibrium state, the point at which simulations performed on different machine can be compared. In order to balance fluxes and reach (an) equilibrium with a complete atmosphere-ocean coupled model hundreds of simulated years may be required, with no guarantee of producing results which are similar to those produced by another model.
In fact, to recover the initial model behaviour on porting to a new machine, it is quite possible that additional parameterization work would be required, for example addressing the top of the atmosphere balance but also to achieve the desired mean states and variabilities of certain quantities (the actual quantities to be addressed with vary for different geographical regions, depending on the topic being studied with the use of the model). The question then is how to validate a port involving simulations containing different parameterizations? This is sue is further discussed in Section 0 in the context of techniques to reduce the sensitivity of the Met Office’s Unified Model to rounding errors.

To avoid such problems and limit CPU consumption during the validation phase, the standard ARPEGE-NEMO coupled model is used in a slightly different configuration. In order to stabilize the system and enable an equilibrium stage that could be comparable on several machines to be reached rapidly, a degree of freedom is removed from the ocean model.

This configuration, called NEMIX[1], has already been used with different ocean models for geophysical studies[2,3]. It consists of a configuration reached by disabling salt and heat transports (1D model) and replacing them by a flux correction. A daily difference between an observed climatology on temperature (salinity) and the corresponding variable calculated on the ocean model is used to deduce heat (water) flux correction necessary to fit observational data. During this first simulation (called “forced”), a daily climatology of flux corrections is performed. At a second stage (called “coupled”), the flux corrections are added to ocean model to reproduce the missing salt and heat transports.

NEMIX Fortran coding details

To be able to easily switch from a conventional ocean-atmosphere coupled model to a configuration using a mixed layer ocean, we modify the NEMO code, but keep the rest of the existing coupled configuration (here ARPEGE-NEMO). The only modification done on atmosphere model ARPEGE is that it is always forced to calculate its own surface albedo and its own ice temperature (namelist values LMCC02 = .T. and LALBEDO=.F.) instead of receiving it from coupling fields: it is a consequence of the LIM ice model disabling on our NEMIX ocean configuration.

Parts of the code like the NEMO advection scheme are bypassed and some instructions are added using a new namelist parameter (nn_cmxl) set to:

* 0 in conventional fully advective mode,
* 1 during the correction calculation (“forced mode”) and
* 2 to use the mixed layer configuration, applying the flux correction (“coupled mode”)

Within the code, to switch from standard ocean to our configuration, we test the value of the nn_cmxl variable within Fortran “IF” conditions. This strategy (instead of pre-processor operation) should allow us, in the future, to separately activate this configuration on AGRIF zoom areas.

No horizontal exchange should be represented: NEMO avoid to calculate and integrate slope of lateral mixing (ldf_slp), advection and diffusion at the bottom layer (tra_bbl), horizontal and vertical advection (tra_adv, dyn_adv), lateral mixing (tra_ldf, dyn_ldf), horizontal gradient of hydrostatic pressure (dyn_hpg) and surface pressure gradient (dyn_spg). The momentum trend is updated with the planetary vorticity trend only (dynvor).

A damping is necessary to avoid systematic drifts: temperature and salinity are damped on the whole globe (nn_hdmp=-1) but only under the mixed layer (nn_zdmp=2). The restoring coefficient varies like an exponent of minus depth. Bottom and surface restoring time scales are set to 30 days, with a transition depth of 800 meters. This damping is activated during both correction calculation and application stages. Damping is disabled at the surface (sbcmod).
During the “coupled” experiment, two new coupling fields of flux correction (heat and water) are received (sbccpl) from a toy model via OASIS. Calculated as described below, the daily climatological value of the heat (water) flux correction is added to non solar heat flux (total E-P balance).

During the “forced” experiment, at the beginning of the time step loop (and not at the end, as usual in a coupled model), the Levitus (or any climatological) temperature (even under ice) and observed corresponding sea ice fraction are sent to the atmosphere (sbccpl). LIM model is disabled and ice cover is forced to observed quantities (sbcice_if). Ice fraction initialization has to be done before coupling (restart reading or calculation regarding Levitus). Fluxes are received after one coupling step from the atmosphere model (sequential coupling). Those fluxes have been calculating according to the climatological SST and Ice Cover values previously sent to the atmosphere model at the beginning of the coupling time step (one day, in our case).

At the end of active tracer update routines, but before density (eos) and tendencies (tranxt) calculations, at each end of a simulated day, tb,tn and sb,sn variable values are re-initialized (in the mixed layer only) with daily Levitus temperature and salinity interpolated at the day following the current day (dtasal, dtatem). Those variables are re-initialized again after tendencies calculations.

At the same stage, the flux corrections are calculated, estimating the difference between total amount of energy (water) brought by the atmosphere during the coupling time step and the necessary energy (water) to drive the model to the Levitus state of the next day. The difference of temperature (salinity) between integrated variable tn (sn) and the Levitus value of the next day is multiplied by rau0.rcp (rau0) and accumulated on the total depth of the mixed layer (determined following the rho criteria).

An offline and highly parallel tool has been developed to process on the parallel supercomputer itself a 365/366 days climatology of the flux corrections, filtering harmonics lower than seasonal. It allows us to follow, during the “forced” simulation, the evolution of the quantities that we will add to the ocean model during the “coupled” one.

An example of porting validation

These flux corrections encompassed three different contributions: (i) the simulation of ocean salt and heat transferts, (ii) the correction of the atmosphere biases and (iii) the correction of the coupling biases.
Consequently, flux corrections are supposed to be suitable to detect a possible anomaly of ocean, atmosphere and coupler behaviour, induced by the change of compiler and running environment.
A good example of anomaly detection is described at paragraph “ARPEGE-NEMIX porting, optimization and performance tests on PRACE tier0-tier1 machines”.

On a first stage, a 30 year long simulation is performed to calculated these corrections. The experiment is long enough to be able to distinguish the effect of a possible error on porting from the interannual-to-decadal atmospheric variability.

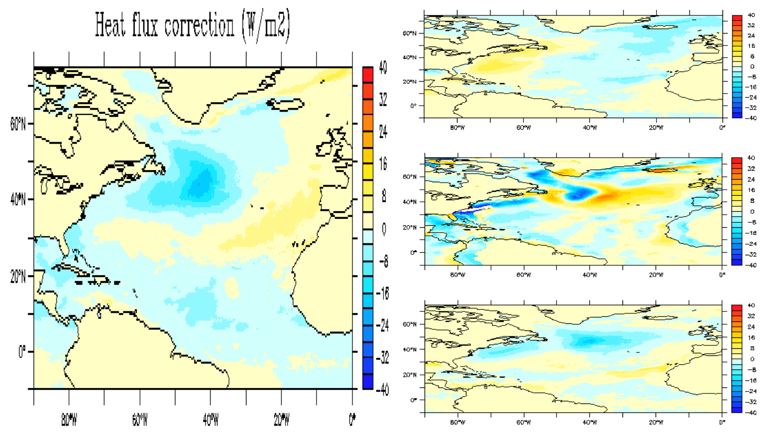


Figure 1: Heat flux correction (W/m2), 30 years mean difference SGI Altix - NEC SX8R (left) and first 3 decadal anomalies (SGI Altix), same range values

The result shown on **figure 1** has been produced comparing 2 identical simulations performed on vector Météo-France NEC SX8R and scalar CINES SGI Altix supercomputers. A regional zoom is done on North Atlantic Ocean, one of the regions where atmosphere strongly influences the mixed layer ocean model variability at interannual time scales. Variations of mixed layer depth is observed in the model, which also implies strong variations of heat flux anomalies.
Differences of heat flux computed on scalar and vector machines (left figure) appear to be comparable to differences between the decadal mean (from top to bottom, years 1-10, 11-20

and 21-30) and the 30 year mean on a single machine (decadal variability). On other regions, differences are one order of magnitude less important, as water flux correction variations. It proves that machine dependant differences do not exceed natural variability.

On a second stage, coupled experiments (using flux correction calculated on the previously described stage) will be performed for further geophysical analyses purpose. Comparing atmosphere and ocean variability on both machines will definitely allow us to reach a conclusion about the portability of the high-resolution coupled ARPEGE-NEMIX climate model on targeted architectures.

Conclusion

Scientifically speaking, NEMIX mixed layer model, if efficient to quickly validate a porting and to lead to interesting mechanism analysis, is not able to address most of the climate modeling community problems. Implementation of the fully advective NEMO model will then be necessary.

For this purpose, NEMO ORCA025-75 vertical levels developed at LEGI laboratory will replace our NEMIX model. Considering similarity of the codes (same NEMO 3.2 version, NEMIX configuration only differs with from NEMO on a few routines), this switch should not be, technically speaking, too problematic but could significantly delay the scientific validation, much difficult to reach because of the higher complexity of the represented coupled phenomena, inducing regional coupled biases, wrong heat balance, etc.
At this stage, the priority is not to enhance the performance of our model (though improved scalability will be required for use of the model on future machines). Due to the increasing complexity of the model, more computer time is required to find parameters that best fit machine characteristics.

ARPEGE-NEMIX porting, optimization and performance tests on PRACE tier0-tier1 machines

Overview

A PRACE “Preparatory Access” project gives us the opportunity to reach several technical objectives on porting, set up on thousands of cores and optimizing the 2-component CGCM ARPEGE-NEMO coupled by OASIS, at high resolution (720 x 360 x 31 for atmosphere, 1442 x 1021 x 46 for ocean) on the CEA TGCC Bullx Curie platform.
Curie Bullx is a scalar parallel supercomputer with a peak performance of 1.5 Pflops, when the full configuration (including thin nodes and accelerators) will be installed at beginning of 2012. The Curie Fat nodes (phase 1) part of machine (that we accessed during our Preparatory access) is composed of 1440 eight-core processors, Intel Nehalem-EX X7560 @ 2.26 GHz total of 11 520 cores.

Performance

The ARPEGE-NEMO climate model, jointly developed by Météo-France, NEMO and CERFACS based on components developed at ECMWF and in the NEMO consortium, has been compiled and run on more than 1000 cores on the PRACE tier-0 “CURIE” Bullx supercomputer.
Our test configuration requires high resolution components (50km-atmosphere, ¼ degree-

ocean), to study regional scale / large scale interactions. The ocean model is used on a 1D mode, as a mixed layer model (as described in chapter “ARPEGE-NEMIX: a simplified high resolution CGCM for validation purpose on PRACE tier-0/tier-1 machines”), to simplify and better understand coupled processes and validate porting more easily (comparing results with those of previously validated simulations on reference supercomputers).

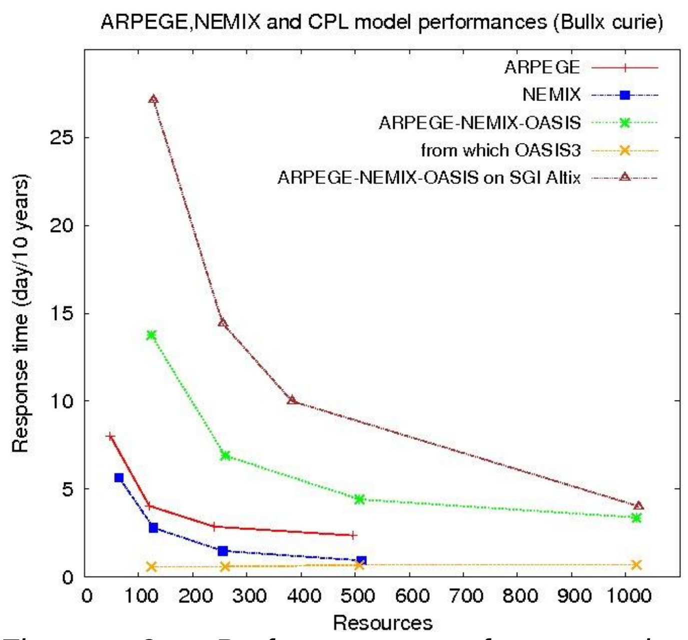


Figure 2: Performances of atmosphere (ARPEGE), ocean (NEMO, mixed layer configuration NEMIX), coupler (OASIS3, always on 12 PE) and whole coupled model (ARPEGE-NEMIX) + comparison with previous ARPEGE-NEMIX performances on SGI Altix

As expected, performances of the coupled system reach 5 days per simulated decades (see **figure 2**), which corresponds to the best results observed on present supercomputers. Except data management problems, this speed allows to comfortably perform decadal (and even centennial) simulations.

A total of 8179 cores has been used without problems (atmosphere:4082, ocean:4096, coupler:1). A higher parallelism could be tested on ocean model, but atmosphere model efficiency seems surprisingly limited to 256/512 cores.

Moreover, the use of a pseudo-parallelized version of our coupler (OASIS3) was still a bottleneck for the coupled system, as 20% of the elapsed time was spend to perform interpolations and ensure communications between coupled components.

Coupling enhancement

To try to overcome this problem, it was necessary to modify ARPEGE and NEMO coupling interfaces to plug the newly developed OASIS3-MCT coupler (see WP4 report on OASIS coupler enhancements). Allowing parallel interpolation of the coupling fields (as a parallel matrix-vector product) and their parallel redistribution directly from the source processes to the target processes , OASIS3-MCT offers an elegant solution to the previous OASIS3 bottleneck.

Interface modifications (from a previously implemented OASIS3 coupling) are very few:

- only « USE mod_prism » instructions differ
- prism_put/get argument arrays size has now to fit exactly prism_define declaration
- namcouple is simplified but has to be modified.

So, generally speaking, it can be considered that a previously developed OASIS3 interface is backward compatible with the new OASIS3-MCT version.
A beta-version of the new coupler has been used on Curie supercomputer and time spent on coupling has been compared, using OASIS3 and OASIS3-MCT couplers. As shown on **figure 3**, slowing down due to coupler process communications bottleneck is basically eliminated, time spent on coupling being reduced to a few tenth of percent. Efficient coupling of high resolution component models with OASIS can now be considered as a solved problem thanks to this new OASIS3-MCT.

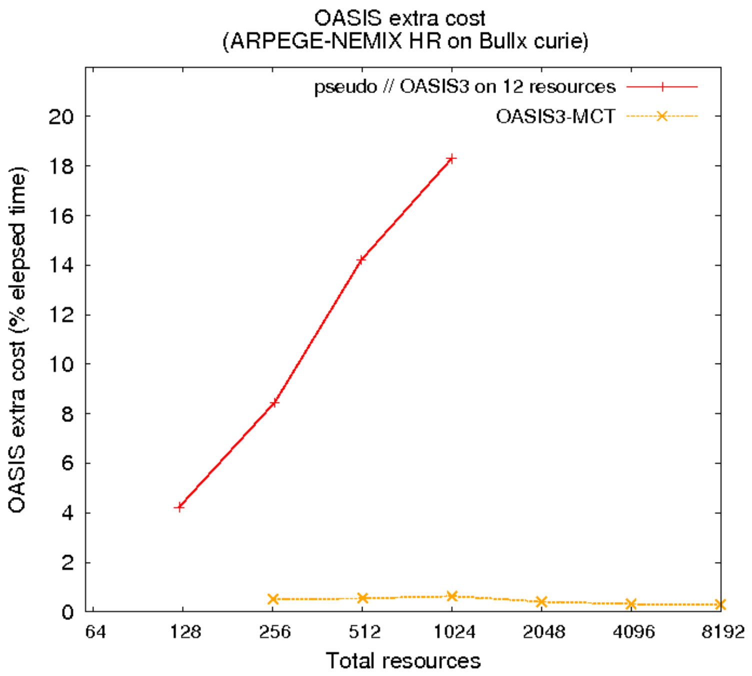


Figure 3: Percent of OASIS coupling elapsed time (communications and interpolations) among total atmosphere-ocean-coupler elapsed time

Porting issues

Coupled model porting has been affected by various problems, mainly concerned with the atmosphere part. As a consequence of the many porting realized for several years on scalar platforms (ECMWF machines, PC, Grids, GENCI supercomputers), our models are now well adapted to Intel compilers. This experience quickens compilation phase.
Nevertheless, problems can still occur when ARPEGE-Climate parallelism increases. High resolution configurations are used in a small number of scientific studies and some features,

such as 2 dimensional partitioning, only necessary on MPP machines, not intensively tested. Some debugging has been done in the code to overcome problems detected during first steps of execution. In particular, a problem in the parallelization of the mass correction routine cormass2.F90 drove a robust bias on surface pressure field over region covered by master processor.
This bias has been detected checking heat flux correction calculated during an ARPEGE-NEMIX forced experiment. **Figure 4** clearly exhibits unrealistic surface pressure anomaly on a geographical area corresponding to master processor related sub-domain.

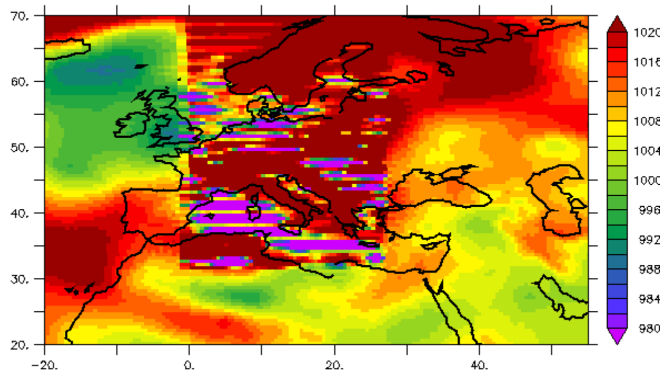


Figure 4: Surface pressure anomaly, hPa, first 6h mean, PRACE tier_0 / NEC SX8R comparison

After correction, a six months long simulation was performed on the machine. Comparing it with similar test experience done on reference NEC SX8R and SGI Altix supercomputers, no important bias could be detected in the results (mean surface pressure and temperature, water/heat flux correction). The mean heat flux correction (mean on 6 months only) is similar to reference.

We consider that our ARPEGE-NEMIX-OASIS3 model is ready to be use at high resolution for a first scientific validation, following the strategy described on chapter “ARPEGE-NEMIX: a simplified high resolution CGCM for validation purpose on PRACE tier-0/tier-1 machines”.
Moreover, further developments will lead us:

1. to fully optimize the various components: different compilation options have to be tested and best performance parameters of the models have to be found (like vectorization length NPROMA for ARPEGE or ideal partitioning for NEMO)
2. to better balance coupling parameters: load balancing between components, mapping (if possible) of different executables on allocated nodes (and node cores)
3. to reduce output data amount, organize data post-processing and migration to local storage disk. This work is crucial for Climate Modeling, considering size of produced data (4Gb/h in our test, much more on production phase)

References

[1] E. Maisonnave, S. Bielli, and C. Cassou. A climate model with a mixed layer based on

nemo v3.2, Technical Report TR/CMGC/10/54, SUC au CERFACS, 2010
[2] C. Cassou, S. Bielli, H. Douville and E. Maisonnave : Limitations of the AMIP protocol to investigate extratropical climate events at seasonal timescale : the 2003 European heatwave as a case study, personal communication
[3] Y.-O. Kwon, C. Deser, and C. Cassou. Coupled atmosphere-mixed layer ocean response to ocean heat flux convergence along the kuroshio current extension. Climate Dynamics, 36:2295,2312, 2011, 10.1007/s00382-010-0764-8

EC-Earth modelling system on MareNostrum Supercomputer: porting and performance experience

Contributors: O. Jorba, K. Serradell, L. Telloli

Objectives

Several versions of the EC-Earth model have been ported on MareNostrum supercomputer. The purpose of the present document is to describe and summarise the main issues related to the portability and performance of the EC-Earth model on the MareNostrum supercomputer. From the experience of this effort, some recommendations are presented to simplify the task of porting a complex Earth System Model like EC-Earth to complex High Performance Computing environments. This report complements the work presented in D8.2 – "Report on the Description of the Evaluation Suite and Base-case Results", chapter 5, "Porting and performance analysis of EC-Earth system on MareNostrum Supercomputer".

A brief description of the MareNostrum supercomputer and the EC-Earth model is presented in Section 2 and 3, respectively. Section 4 describes the porting issues encountered during the implementation of several versions of the EC-Earth model on the MareNostrum supercomputer. A trace analysis of an EC-Earth execution is presented in Section 5, focused on the identification of those parts of the code that show some possible performance improvements in the future.

HPC environment: the MareNostrum supercomputer

The MareNostrum supercomputer is a high performance computing facility hosted by the Barcelona Supercomputing Center (BSC). It was built in March 2004 and has been upgraded in one occasion. The next upgrade is planned for 2012. The supercomputer is a node of DEISA2 and it is included in PRACE initiative as a Tier-1 machine. With the next upgrade, it will be included in Tier-0 infrastructures.

The main technical characteristics of the system are described in detail in the following link: http://www.bsc.es/marenostrum-support-services/marenostrum-system-architecture/. A brief description of the infrastructure is presented here.

The MareNostrum supercomputer is based on processors PowerPC, architecture BladeCenter, Linux operating system and Myrinet interconnection. MareNostrum has a total of 10240 IBM Power PC 970MP processors. The Peak Performance of the system is 94.21 Teraflops. Each Blade Center has 14 server blades type JS21. Each of these nodes has 2 PowerPC 970MP processors (each processor has two cores and note that cores are referred to as CPUs in this section) running at 2.3 GHz and 8 Gb of shared memory. MareNostrum has 20 storage servers arranged in 7 racks that work 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/marenostrum-support-services/available-software).

The EC-Earth model

The EC-Earth model, European Community Earth system model, is one of the Earth System Models contributing to the Coupled Model Intercomparison Project (CMIP-5). It is developed by several European National Weather Services and research groups. It is based on the seasonal prediction system of European Centre for Medium-Range Weather Forecasts

(ECMWF) [Hazelenger et al, BAMS].
The modelling system consists of two main components and a coupler: 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 atmospheric component is based on the Integrated Forecasting System (IFS) of ECMWF. The basic configuration contains 62 levels in the vertical, and an horizontal grid of 1.125 degrees spacing (125 km). The chemistry of the model is based on the TM5 module, which is included online in the IFS model. The land-vegetation model is the Tiled ECMWF Surface Scheme for Exchange processes over Land (HTESSEL) and it is part of the atmosphere model.
The ocean and sea-ice model used within EC-Earth is NEMO (Nucleus for European Modelling of the Ocean). The approximate resolution of the ocean model is 1 degree with a refinement in the southern pole. It uses 31 vertical layers. The sea-ice model is the Louvain-la-Neuve sea-Ice Model (LIM).
The EC-Earth has been ported 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 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 2.0: Coupled model (IFS CY31R1– OASIS3 v2.5 – NEMO2/LIM2), with local ECMWF modifications and EC-Earth developments.
- Version 2.2: CMIP5-ready coupled model (IFS CY31R1– OASIS3 v2.5 – NEMO2/LIM2)
- Version 2.3: updated version 2.2
- Version 3.0: Coupled model (IFS CY33 – OASIS3 v2.5 –NEMO3/LIM3) – beta subversions

Porting experience of EC-Earth on MareNostrum

Several versions of EC-Earth have been ported (compiled and run) in MareNostrum: version 2.0, 2.1, 2.2 and 2.3. Detailed instructions for compiling and running the model in its version 2.0 are found in Stefanescu, 2008. However, significant extra effort was required to compile the model on MareNostrum. In this section, we present the main efforts undertaken to port the model to MareNostrum system, focusing on: *machine access, compilation and build, setting up and model run, debugging and testing, input and output file management*.
The EC-Earth model is programmed in Fortran and C languages and uses some external libraries for mathematical, I/O and parallelisation issues. Prior to any porting effort, several packages are required in the targeted HPC system (ksh, perl, mpi1, fortran90-95 compilers, openMP and netCDF). All the required packages are widely used and did not represent an issue to finding or porting them on MareNostrum.
MareNostrum is an IBM machine that runs with Linux operating system. This may cause some problems when porting codes developed on other architectures. In this sense, some

specific settings for the MareNostrum architecture are needed, and a search task has to be done prior any successful compilation of the code.
The porting of the EC-Earth model was complex and took a large amount of time for versions 1, 2, subversions 2, but importantly improved in version 3. Here we describe the main issues related with version 2.3, and how most of them improved in version 3. From all the components of the model, the main porting issues were related with the IFS atmospheric component. The OASIS and NEMO portings were straightforward (specification of MareNostrum compiler environment and options are included in the main configuration file). Thus, the following description mainly focuses on the IFS porting.

Machine access:
The access to MareNostrum system is done through SSH connections. It is a standard on DEISA and PRACE systems, and it is not considered a specific problem. The limitation may be found in the band-with connection to the system. The MareNostrum uses a gigabit network connection for the Input/Output of information to the system. This allows a high transfer rate, but the user connection network becomes the limitation. It is usual that user experience some problems downloading the datasets from MareNostrum. The large amount of data generated in the system is not easy to handle, and specific strategies for the movement of datasets should be designed prior any execution effort.

Compilation and build:
Some minor modification in the source code of IFS were needed to compile versions 2 and subversions. The code include some #ifdef statements specific for AIX, Cray and other architectures. Due to the particularities of MareNostrum (hybrid IBM and Linux environment) a specific case was included in the code.
The recommended compilers in MareNostrum are IBM XL compilers. This type of compilers do not include by default the underscoring in the Fortran routines, which it was required by the C code that calls Fortran routines in the original EC-Earth code. Thus, several routines need a specific forcing of the underscore by the compiler. This is done through the compiler flag –qextname. Several routines need this procedure:
qextname=utdec,utopen,utinv,utfree,utmake,utdiv,etime,dtime,utmult,utscal,utenc,utcut,flush,utcvt,utexp,utorigin,uttime,utcaltime,utcpy,flush,abor1,follow,addrdiff,broadcint,broadcreal,c_drhook_init_signals,c_drhook_print,dr_hook_util,dr_hook_util_multi,ec_getenv,ec_mpi_atexit,ecmwf_transfer,ec_numenv,ec_putenv,ecqsort,ec_raise,ec_strenv,fft992,getcurheap,getmaxstk,get_opt,getpag,getrss,getstk,gstats,gstats,follow,gstats_label,gstats_label,follow,gstats_print,gstats_psut,gstats_setup,hostnm,ifssig,ifssigb,isrcheq,isrcheq,follow,isrchfge,isrchflt,minv,minv,follow,mxmaop,mxmaop,follow,profile_heap_get,rdot,rg,rsort32_func,rsort64,rsum,set99,sgtsl,sigmaster,user_clock,bubox,bucrkey,bufren,bufrex,bunexs,bunpck,bunpks,bupkey,buprq,buprs0,buprs1,buprs2,buprs3,buprt,buprtbox,bus012,busel,busrq,buukey,fmmh,pbbufr,setlalo,c_drhook_start,c_drhook_end,abor1fl,cdrhookinit,cdrhooksetlhook,drhookprocinfo,drhookprt,c_drhook_set_lhook,c_drhook_init,dr_hook_procinfo,dr_hook_procinfo,irtc_rate,dr_hook_prt,dr_hook_prt
In addition to the changes described above, a particular compilation for IFS was needed for versions 2.1, 2.2 and 2.3 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.
The structure of compilation files included in IFS is rather complex and this is therefore also

the case for EC-Earth. There exists a general compilation file for EC-Earth where the environment variables and some compilation flags are defined. However, several modules of IFS also contain a specific configuration file that needs some adjustment. This results with an arduous task of identifying the configure file that needs modifications. Some of the modules need some specific compilation flags that should be defined properly to succeed with the porting.
Once a successful porting of the model is obtained, the porting of a next version is more straightforward, but there are always some routines or parts of the code that require specific attention.

Setting up and model run:
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 (i.e. cores) 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 two dual core processors resulting in a total of 4 CPUs (or cores) 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 that required on other platforms.

Debugging and testing:
The domain decomposition for IFS and NEMO is defined in the compilation files. A specific executable is then associated to a specific domain decomposition and number of processors. If the user is performing some test or scalability studies, the code must be recompiled. This may be overcome if the domain decomposition is included through a namelist file or similar, instead of defining it in the compilation file.
Totalview, Paraver, Dimemas software tools have been used to debug and test the EC-Earth model in MareNostrum. Some specific tasks related with the analysis tools were undertaken, but did not interfere with the EC-Earth porting.

Input and output file management:
No problems were encountered in the management of Input/Output of the EC-Earth model. The GPFS file system of MareNostrum performed well with the EC-Earth I/O design. For future applications, a more advanced I/O approach should be implemented within EC-Earth in order to take advantage of parallel I/O.

Most of the problems encountered with version 2.0, 2.1, 2.2 and 2.3 were overcome in beta versions 3.0. A specific parser has been developed in versions 3.0 of the EC-Earth model. Now, there exists a general configuration file, an XML file, where all the compilation options for all the modules and submodules are specified. This strongly simplifies the porting task of the model in MareNostrum. There are still some files with #ifdef statements that should be checked carefully and adapted to the MareNostrum architecture characteristics.
With the experience acquired in the porting of EC-Earth, it can be said that there aren’t major

problems in the programmability of the system or the coupling structure. Some old and inherited complex structure of the coupled system hampers the porting exercise, but this has been circumvented in version 3.

Performance of the model on MareNostrum: trace analysis

The Paraver software is used to analyse the performance of the EC-Earth model in MareNostrum. Paraver is an open source performance visualization and analysis tool developed in BSC. Figure 1 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, 100 for IFS and 32 for NEMO in our example). Paraver provides several time statistics for analyses.

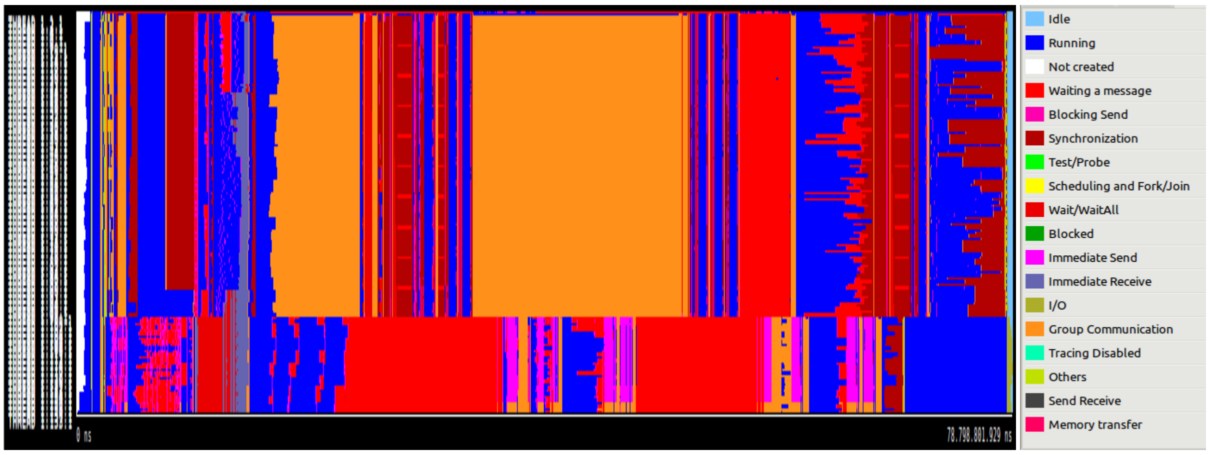


Figure 1: Example of a Paraver graphical view. OASIS, IFS and NEMO are run with 1, 100 and 32 CPUs, respectively. The X-axis represents time and the Y-axis represents the different CPUs (1 for OASIS, 100 for IFS and 32 for NEMO in this example).

In this Section, a raw-trace analysis of an EC-Earth run is discussed. The model run is configured with 1 CPU for OASIS coupler, 100 CPUs for IFS atmospheric model and 32 CPUs for NEMO ocean model. The discussion focuses on the communication pattern of the atmosphere and the ocean models and some parts of the code are selected to exemplify load balance problems between processors and serialization patterns that may be optimized by analysing the code in detail. No optimizations are presented in this report.

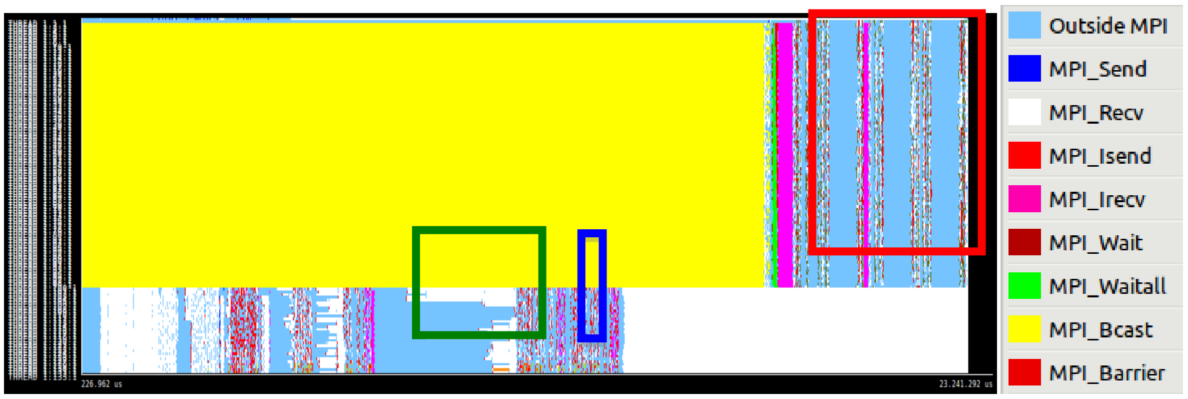


Figure 2: Visualization of 3-hour run. The red square indicates the zoom in the IFS execution shown in Figure 4, the blue square the zoom in the NEMO execution shown in Figure 5 and

the green square the zoom shown in Figure 6.

Figure 2 shows a visualization zoom of a 3-hour run. The first row of the image presents the activity of the coupler. The white colour represents a MPI_receive state, meaning that the coupler is waiting the end of an execution of the atmosphere and ocean (communications every 3 h). Once the atmosphere is ready to send information to the coupler, the white colour changes to blue, indicating that the coupler starts its computation work. The central part of the image, rows 2-101 represent the atmospheric model execution. The light blue represents computation while the other colours are MPI states. The distribution of execution shows that IFS uses several MPI_Bcast to structure the execution of the code. On the other hand, rows 102-134 shows the CPU state for the ocean execution, where no MPI_Bcast calls are identified. For parallel applications the latter approach is preferred if the application is well load balanced. In the analysed configuration, the wall-clock time of an IFS and NEMO execution and coupling are not balanced, thus the CPUs of NEMO are waiting IFS end execution. It is important to note from this trace, that IFS is spending a long time in a broadcast (MPI_Bcast). Such behaviour can be attributed to a poor performance of the interconnection network of MareNostrum, that under some conditions of overloading performance is lost.
The communication pattern between CPUs will provide useful information on the performance of the model. In highly parallelised codes, a key issue is the limitation of communication between nodes. Figure 3 shows the communication pattern of the EC-Earth model.

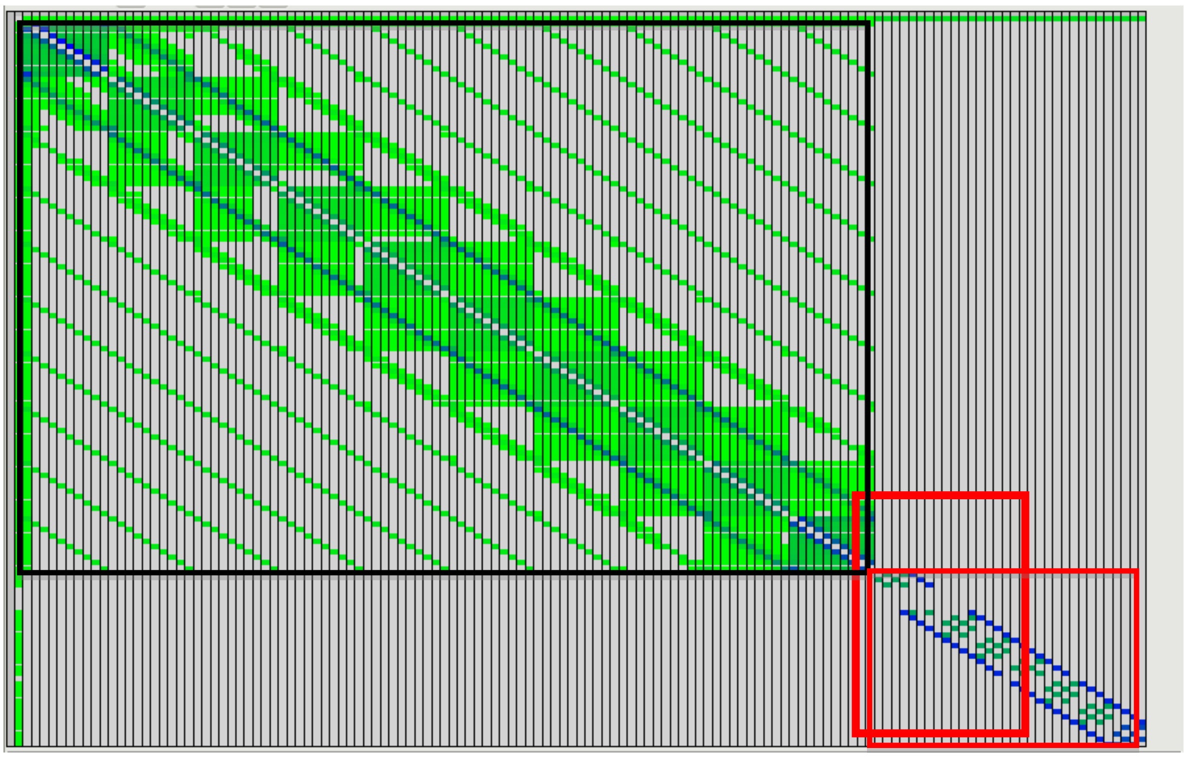


Figure 3: Communication pattern of EC-Earth run (Green: low amount of bytes sent between processes; dark-green: medium amount; blue: large amount). Black square limits IFS processes, and red square limits NEMO processes.

In this visualisation, the total bytes sent between processes are displayed (light green: low amount of communication; dark green: medium amount of communication; blue: large

amount of communication) in an array structure, where x-axis and y-axis are the processes. The diagonal represents the communication of process n with process n, and thus, there is no transfer, outside the diagonal the communication of process n with process m is displayed in colours. The structure of the model is well identified in the image. The coupler communicates homogeneously with all processes of the atmosphere and the ocean. The communication pattern for IFS (black square) is more complex than for NEMO (red square).
In IFS, a rather complex pattern of communication is identified between processes. During an iteration, a process communicates with a group of 20 neighbour processors, and with 6 processors away from it. More communication is detected in the poles, where a group of 10 processes interchange larger amount of data than in the centre of the domain. Additionally, communication with the 10th neighbour processor is larger than with the prior 9.
Regarding NEMO communications, each process communicates with its 4 neighbours. There are specific processors that exchange a large amount of bytes (blue colour), creating a group of communications every 4 processors. Such structure is characteristic of finite difference partial differential equation models, where the domain discretization is done in the x-y plane. It is also important to note that there is no master process communicating with all the CPUs of NEMO. This implementation benefits the reduction of communication among processors.
Overall, Table 1 summarizes the average time spend by processes in computation and different communication states (specific MPI call). The statistics are obtained from all the trace presented in Figure 1. Results show how the MPI_Recv state is dominant in the coupler and ocean modules, while the atmosphere spends most part of the time in an MPI Broadcast. We believe that such performance is attributed to a low performance of the interconnection network of the supercomputer. The coupler spends most of the time waiting for the model results of the ocean and atmosphere. The ocean spends 34.26% of the time computing and 62.91 receiving information from the coupler. The statistics of the atmosphere are strongly affected by the weight of the Broadcast, however it is important to note that 2.34% of the time the processes are in an MPI_Barrier. The computation time of IFS represents 15.16% of the total time of the trace. Note that the trace includes initialization time.

	coupler	atmosphere	ocean
Outside MPI	23,33	15,16	34,26
MPI Send	0,22	0,01	0,00
MPI Recv	76,45	3,50	62,91
MPI Isend	0,00	0,00	0,39
MPI Wait	0,00	0,00	0,10
MPI Waitall	0,00	0,52	0,00
MPI Bcast	0,00	76,42	0,00
MPI Barrier	0,00	2,34	0,00
MPI Allreduce	0,00	2,06	1,72
MPI Gather	0,00	0,00	0,23
MPI Gatherv	0,00	0,00	0,00
MPI Scatter	0,00	0,00	0,40

Table 1: Average time (%) spend by processes in computation and MPI states.

With the aim of showing some examples of load balancing and serialization issues within IFS and NEMO, we present a visualization zoom of some parts of the IFS execution and then some parts of the NEMO code. Note that Paraver allows the instrumentation of both user code routines as well as MPI routines. Figure 4 shows a zoom within the IFS execution where the 3 hour run cycle can be clearly identified with a repetition pattern. From this zoom, we identified two examples of load imbalance in subroutines *slcomm2a* and *trtgtol* (light green,

green). The processes spend a significant time in both routines. From Figure 4 the load balance problem of *trtgtol* routine is clearly appreciated, but not in *slcomm2a*. Figure 5 shows a zoom cantered in the zone where the processes are executing the *slcomm2a* routine.

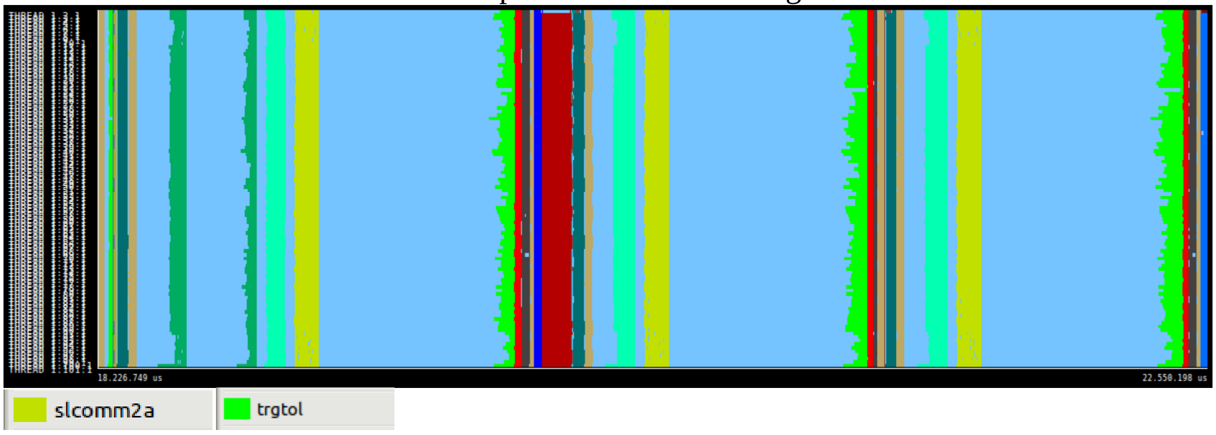
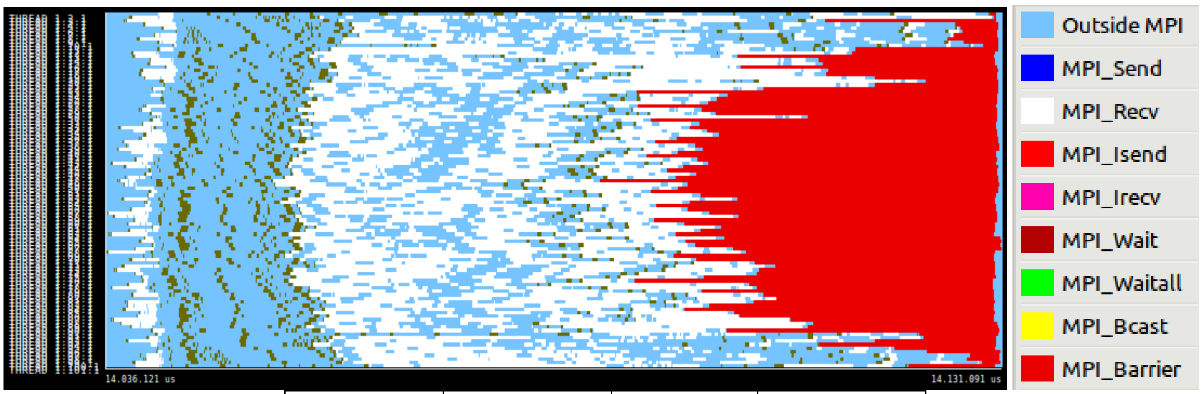


Figure 4: Visualization zoom of the IFS execution subroutines. Three hour run cycles are identified. Figure 2 displays the location of the zoom. Subroutines slcomm2a and trtgtol colours are displayed in the bottom of the figure.

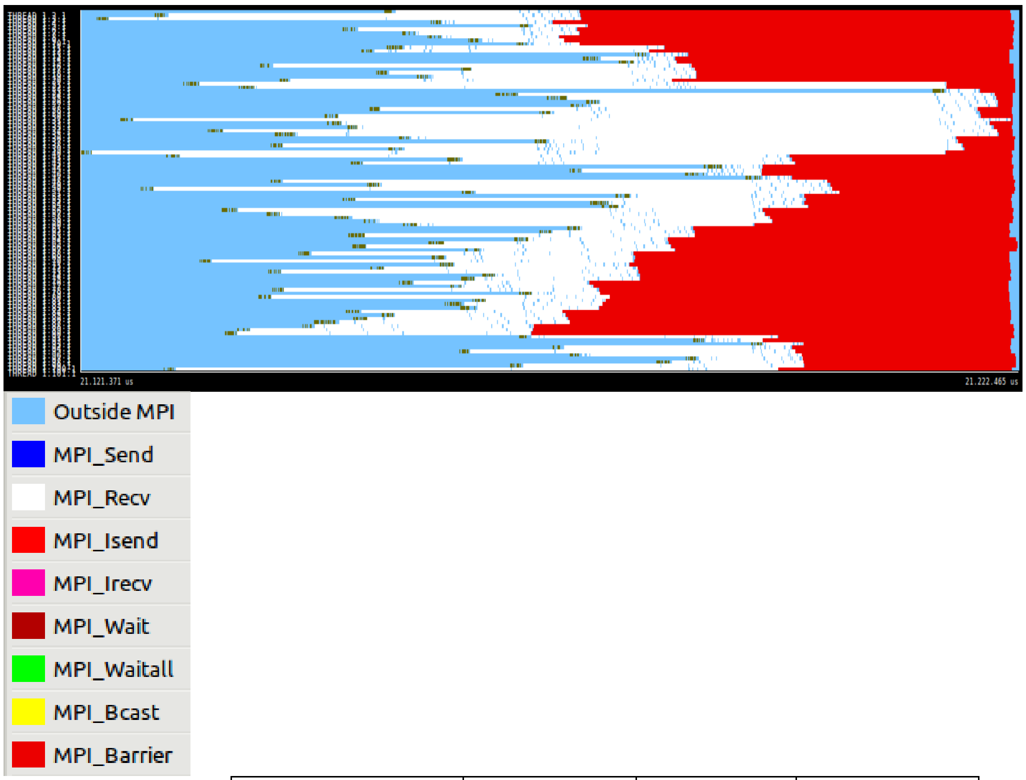
The visualization of Figure 5 is configured to show the MPI states of the execution of the routine *slcomm2a*. The pattern of the trace clearly shows important load balancing problem among processors. On the average, the processors are computing 39.08% of the time, receiving information 34.05% and waiting in an MPI_Barrier 26.87% of the time. However, the maximum time in computation is a 70.50% of the total time and the minimum time represents 28.63%. This means, that there are processes that only spend 30% of the time computing, while others spend 70%. This indicates a low parallel performance, with a ratio parallelization of 0.55 in the computation time, 0.61 in the receiving time and 0.58 in the barrier (the ideal ratio of parallelization is 1 for perfect parallel codes). These statistics clearly indicate that within the *slcomm2a* subroutine exists large imbalances that may be improved by examining the code in the communication layer.



	Outside MPI	MPI_Recv	MPI_Barrier
Average	39.08%	34.05%	26.87%
Maximum	70.50%	56.27%	46.35%
Minimum	28.63%	20.62%	0.06%
Avg/Max	0.55	0.61	0.58

Figure 5: Zoom of the code execution of the slcomm2a subroutine of IFS. Colour legend at the right hand side of the figure. Outside MPI means computing time.

A second example of load balance problems is identified in *trgtol* subroutine (Figure 2 green square, Figure 4 green colour and Figure 6). Figure 6 visualizes the MPI states within the *trgtol* routine. Again, we can clearly identify processes that spend a long time in the MPI_Recv state, others with larger computation cost, and a large number of processors spending near half of the total time in an MPI_Barrier. The parallelization ratio remains below 0.6 for computation, MPI_Recv, and MPI_Barrier. This indicates that the processes are not well balanced. Such behaviour can be inherent to the problem that the routine is solving, or in some cases is an indication of some program deficiencies. Further work would be oriented in analysing the routine and identifying the reasons why some processes spend such large time in receiving information or waiting in the barrier. In parallel codes, it is recommended not to use a large number of barrier calls. A well-balanced code may not need the use of any barrier.



	Outside MPI	MPI_Recv	MPI_Barrier
Average	38.25%	32.85%	28.90%
Maximum	93.42%	93.54%	51.86%
Minimum	2.10%	2.33%	0.04%
Avg/Max	0.41	0.35	0.56

Figure 6: Zoom of the code execution of the trgtol subroutine of IFS. Colour legend at the right hand side of the figure. Outside MPI means computing time.

Finally, we present an example of serialization encountered in the NEMO ocean execution of EC-Earth. Figure 7 presents a visualization centred over the 32 NEMO processors. The upper panel shows the MPI callers from where we can identify what part of the code is in execution by each processor. A clear serialization pattern appears in this view. Such serialization will produce a delay in the code, more relevant when we increase the number of processors executing NEMO. In this sense, this is a clear candidate to improve for future applications of the model with a large number of processors. The routine involved in this part of the code is the *mpp_lnk_3d*. It appears at first glance that the processors communicate in groups of 4. To corroborate that, the middle panel of Figure 7 displays the MPI states and the communications between processors. A zoom of the first 8 processors is presented in the bottom panel of Figure 7. From such views, one can identify the communication pattern. There is a master processor that sends information to next three processors and after some computation time sends and receives information from the other processors. After that, the masters communicate with the next group of 4 cpus, sending some information. This structure is repeated among all the processors. After analysing the routine involved in such communication pattern it could be possible to provide an optimization approach where the communication patter could be simplified and the staircase structure could be strongly converted to a more parallel pattern.

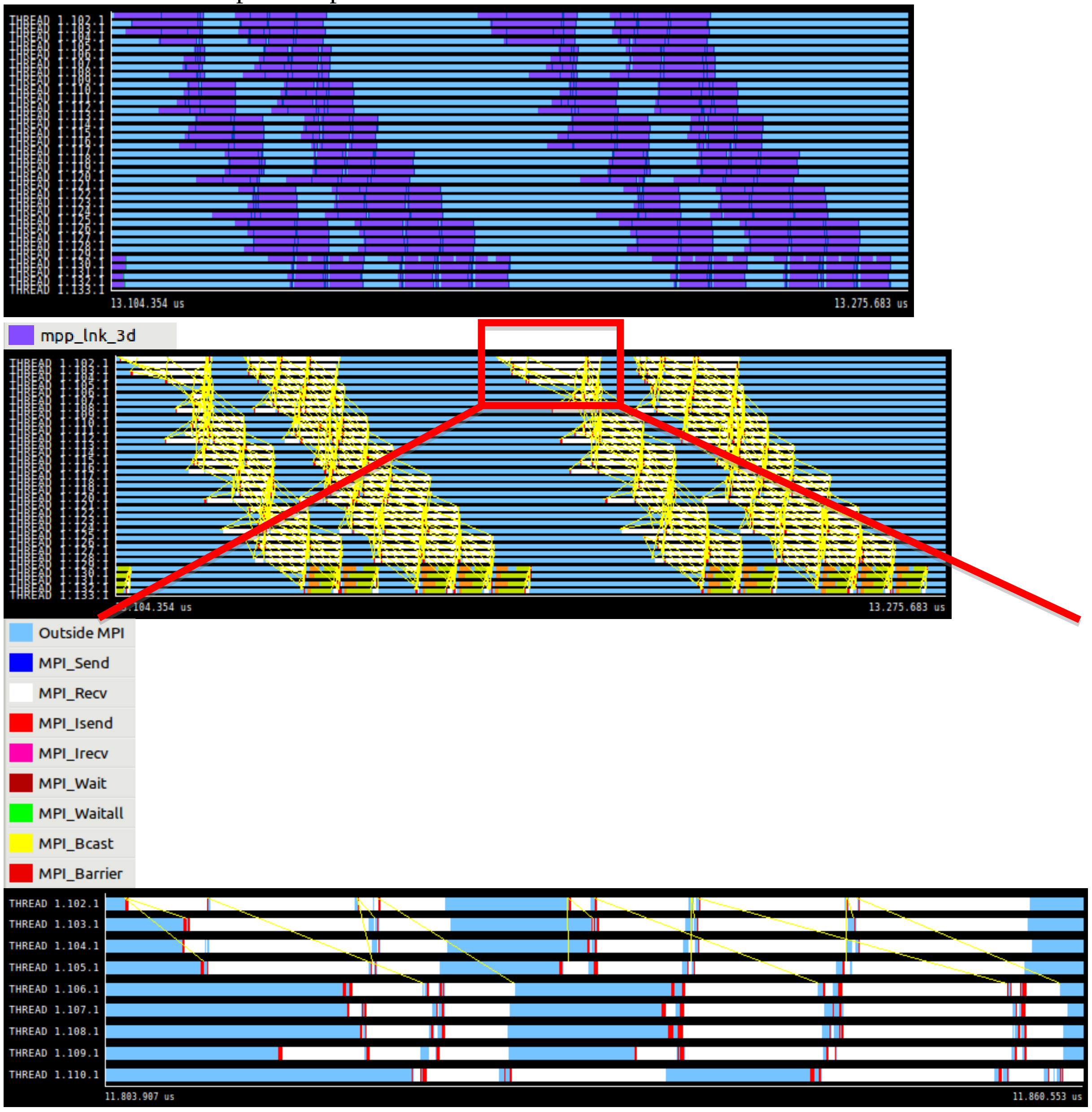


Figure 7: Visualization of an execution part of the NEMO model. Upper panel: visualization of routine executions (violet colour: mpp_lnk_3d routine); middle panel: same zoom showing MPI states and communications between processors; bottom panel: zoom of middle panel view of 9 processors).

The trace analysis is a powerful tool that help identifying some parts of the code that have a poor parallelization degree. From a starting point where we do not have any information about the code and how it performs in a highly parallelized environment, the Paraver software easily contribute in identifying those parts of the code where serialization or load balance problems occurs. After a first inspection of the trace, we identify those parts of the code that are repeated in the time cycle and with several visualization layers and zooms one can search for communication patterns or structures that are candidates to be improved for a better efficiency of the code. It is important to note that the same code will behave different under different architectures. In our results, the communication network of MareNostrum strongly impacts in the code performance with delays in some communication levels. Overall, we have shown a couple of routines where some improvements could strongly impact on the degree of parallelization of the code.

Summary and conclusions

Porting an ESM on a new architecture is a complex task. The experience of porting the EC-Earth model on MareNostrum supercomputer has provided some insights on how could this process be simplified. An ESM is a metamodel with several modules and dependent libraries. Some standard libraries are usually installed in the target architecture. In the MareNostrum case, the NetCDF, openMP, MPI were already implemented. Most of the problems encountered during the compilation of EC-Earth were related with the IFS model. It is important to note that nor the NEMO model neither the OASIS coupler were difficult to compile in MareNostrum. Thus, the layer of configuration and compilation of an application is of critical relevance in a porting exercise. The main difference between IFS and NEMO or OASIS is that in versions v2 of EC-Earth, the configuration files of IFS were not centralized, and several files located in different directories required special attention prior the compilation. Furthermore, not all parts of the IFS demand the same compilation options. This makes the compilation process difficult. In recent versions of the model, EC-Earth v3, the code incorporates a build environment that strongly simplifies the porting process. Now, only a unified configuration file contains all the required information for the compilation.

Thus, a recommendation would be directed to porting ESM models or toy-versions of the models in most supercomputer platforms at European level. The experience acquired in a porting exercise of a model should be maintained within the model structure. In this sense, the experience of widely spread community codes like WRF or CMAQ indicate the way to follow within the climate community. An extensive exercise of porting the codes on several architectures would provide the details required to strongly simplify the installation of such complex codes in new architectures, where several configuration options would provide the best starting point for porting the model. Moreover, if several libraries and applications are already implemented in HPC centres (NetCDF, Blas, Lapack, oasis, nemo), a porting exercise would be much affordable that it is nowadays.

Several versions of the EC-Earth model were successfully ported to the BSC MareNostrum supercomputer, a system based on IBM PowerPC 970MP processors and run under a Linux Suse distribution.

The EC-Earth performance was analysed with respect to trace analysis with the Paraver software. Examples of load imbalance and serialisation within IFS and NEMO models have been presented and discussed. Several options appear for performance improvement of the codes. However, the current numerical approaches of the model, spectral transform regular lat-lon grid in the case of the IFS strongly limits its scalability, and a new numerical solutions are required to improve the system. In this sense, finite-volume, spectral elements, new decomposition grids appear as new options from where current models should evolve.

Acknowledgments:

We wish to thank Xavier Abellán from the BSC Operations team 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.

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].

High Resolution EC Earth Porting, Benchmarking on CURIE

Contributors: Chandan Basu[1]
[1] National Supercomputer Centre, Linköping University, Sweden (LIU)

Introduction

The EC-EARTH[1] is an earth system modeling application. The three main components of EC-EARTH are IFS (for atmosphere), NEMO (for ocean) and OASIS (for coupling). We have benchmarked a high resolution version of the EC-EARTH configuration for scaling. The ocean component used for this configuration is ORCA025 which is a 1/4° global configuration. The atmosphere component uses T799 resolution. The source code of NEMO is v-3.2, the IFS version is Cycle 36r1 and the coupler is OASIS3. For our benchmarking and porting exercises the source code of EC-EARTH, with necessary modifications, input files and run scripts is obtained from SMHI[2]. The system used for our scaling studies is Curie[3] which is a petaflop machine in Europe. The work is also part of a PRACE project on the petascaling of EC-EARTH (work packages 7.2 & 7.5) in which we are involved. The performance of any MPI program may depend on various components, e.g., compilers, MPI libraries, network, file system, processor technologies etc. As EC-EARTH is a large and complex program its porting on a new machine is a considerable challenge. After porting to a new machine, the performance and scaling of the code needs to be evaluated. The goal is to figure out the bottlenecks and to look at suitable strategies to improve the scaling. To do our porting and scaling tests on Curie we follow a work flow which makes our porting work systematic. The same work flow can be followed in other systems for a better porting experience.

Test system : The hardware and the software of Curie system used by us is given in Table 1.

Table 1

System	Curie
Processor	Intel(R) Xeon(R) CPU X7560 @ 2.27GHz

1 http://ecearth.knmi.nl/

2 Swedish Meteorological and Hydrological Institute

3 http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm

Interconnect	InfiniBand QDR Full Fat Tree network
Node	32 cpu cores
MPI processes / node	32
global filesystem	Lustre
Compiler	Intel compiler v-12.1.1.256
MPI	bullxmpi v-1.1.8.1

Work flow for Porting

In modern supercomputers often environments are different from one system to another. So porting a complex code on a new system is always a formidable challenge. Often large codes try to tackle this problem in terms of different per-processor flags to choose compiler, parallel / serial compilation etc. While this approach addresses the problem to some extent, still each system poses its own unique issues. In our porting work we have tried to create an identical environment across different systems for more convenient porting experience. We will describe our work-flow here.

- Basic work flow : We have created a simple configuration script where we define all the environment and the repeatable steps using command aliases and scripts. Also in this configuration script one has to enter some information, e.g., work directory, run directory etc. These settings are then written to an environment file by the configuration script. When screen session is launched by the configuration script these settings are read and loaded in the shell. For each task / activity the configuration script will :Note examples of all script files etc. are available on request from the author.

1.1 Start a screen session with multiple child screens. Screen is a basic linux utility. Using screen utility one can manage multiple windows efficiently. This is useful as often it is needed to have multiple windows open. Moreover screen sessions can be resumed even if the remote connection is snapped due to some reason.

1.2 Define a set of useful command aliases, scripts, environments relevant to a job: These aliases are available on all child screens launched by the configuration file.

1.3 Each work / activity runs under its named screen window launched by the configuration file. Multiple work / activity can be run simultaneously by launching multiple different configuration files with unique environment for each, e.g., one configuration can load OpenMPI while another configuration can load Intel's impi and both can be run simultaneously without affecting each other.

1.4 This is also useful if for porting or benchmarking purpose we locally install test software, e.g., some MPI. For such test installations normally job schedulers will not export all the necessary paths. Through our configuration settings we can export all these paths easily to run jobs on cluster in such cases.

2. Starting a sample configuration called “test” on Curie with multiple child screens :

2.1 mkdir -p ~/config. This is the root directory for storing configuration files. The main configuration file called “test” is located here.

2.2 mkdir -p ~/config/scripts/test. All the other scripts related to the configuration “test” are kept here

2.3 mkdir -p ~/config/comment. Folder for keeping notes, comments. For the configuration “test” a file ~/config/comment/test will be created automatically.

2.4 In the ~/config folder create the configuration file called test.. In the ~/config/test file changes that may be made are in a block marked EDIT where suitable values may be sepcified.

2.5 In the file ~/config/test we have defined few frequently used command sequences through aliases.

2.6 Some aliases require some other scripts. These commonly used scripts are defined in a script called ~/config/scripts/test.

2.7 chmod +x ~/config/test

2.8 Create a sample .screenrc file in ~/config

2.9 Append the .bashrc file with the following lines :

```
if [ -f ~/config/env ]; then
x=`find ~/config/env -mmin -1`
if [ -n "$x" ]; then
source ~/config/env
fi
fi
```

2.10 These lines will not affect the .bashrc behavior except for the next step

2.11 Run ~/config/test. This will create / resume 5 child screens named test. To jump to a particular window press Ctrl-a n where n is the window number.

2.12 Once the screen session is started start working under that environment. Typically this starts with copying the source code related to work to the root work directory as defined in the ~/config/test file.

2.13 The paths / aliases / scripts defined / pointed in test file are available to all the child screens. So for this example on any screen window named “test” :

1.1.1 pressing b will change directory to root work directory.

1.1.2 Pressing c will open a file ~/config/comment/test appending a current date stamp. This is convenient for keeping notes

1.1.3 Pressing t will create a backup of root work directory named with time-stamp. Some rules for inclusion / exclusion of files are already defined in the files backup.sh and exclude.txt which can be modified according to the requirement.

1.1.4 Pressing s will will open a dummy job script and upon closing this job will submitted to the scheduler.

1.1.5 Pressing j will show the job statuses.

1.1.6 intel compiler version 12.1.7.256 and bullxmpi version 1.1.8.1 will be available

2.14 If the screen session is disconnected due to some reason the process running under screen will not die. The processes running under screen can again be accessed by doing step 11 above.

For actual work the configuration script ~/config/test should be given a relevant name. The configuration script's relevant portions should be modified to suit the work requirement. All the directory structures will be created by the configuration script itself but other work specific script files, e.g., backup.sh, run.sh etc. have to be manually copied or created. This is done only once. All the frequently used command sequences can be easily defined this way. This comes very handy while working. One need not have to think about loading correct modules / libraries / binaries in each terminal window as they are opened with correct settings. Also different configuration files can be created for trying different things simultaneously. In porting work often it is needed to try different compilers. So for each compiler one configuration file can be created. Both the configurations can be run simultaneously. Moreover if this configuration is moved to another system then the relevant portions in the configuration file can be modified to suit that system. After launching the configuration file in the new system it will give similar environment as the old system.

Compiling EC-EARTH on Curie

We describe below the steps of EC-EARTH compilation:

1. The configuration file used for compiling / running the hres version of EC-Earth is called ~/config/ecearth_hres.(Specific configuration files are available on request to the authors). The compiler, mkl, mpi and other libraries needed for the compilation are defined here.
2. The important thing in the EC-EARTH compilation is to set up an appropriate xml file with description of system specific settings for package paths, compilers, flags, libaries etc. This file is ~/config/scripts/ecearth_hres/curie.xml. In this file paths for blas, netcdf etc. are specified in accordance with the modules that are defined in ~/config/ecearth_hres
3. Start the configuration by running

$~/config/ecearth_hres

4. Copy IFS, NEMO and OASIS source code and other related scripts in the root folder defined in the configuration file ~/config/ecearth_hres
5. In any of the screen windows run clean command by pressing c. The clean command c is defined in the configuration file ~/config/ecearth_hres. This cleans all the previous installations if any.
6. Press m to start make of EC-EARTH. This first opens a compilation script ~/config/scripts/ecearth_hres/build.sh. Here one can choose which part of the ec-earth to compile. Upon closing file the actual compilation will start. As the compiler, netcdf, blas and mpi libraries are already preloaded in the environment it is not needed to load them any more.

Running high resolution EC-EARTH

EC-EARTH compilation produces multiple binaries, one each for NEMO, OASIS and IFS. The bullxmpi that is available on curie is capable of launching MPMD runs. Moreover an accurate placement of binaries is important for better performance.

EC-EARTH run requires a large number of input files. The input files are namelist files or data files. The namelist type files are input files describing different parameter values which control the flow of run. The data files generally contain initial / boundary / restart values obtained from some previous runs or from some other experiments. The data files are stored in a common place and accessed from there.

The EC-EARTH run is started by a run script which is given to the scheduler. In our porting / scaling experiments we need to do a large number of test runs. We try to do it in a way so that we can analyze the timing results in systematic way.

We do all our work under the ecearth_hres configuration which is launched by the script ~/config/ecearth_hres. Under this configuration to submit a job one needs to press s. The command s is defined in the configuration file ~/config/ecearth_hres. This command executes the following steps sequentially:

- open a file called ~/config/scripts/ecearth_hres/wr.sh. In this file the cpu cores required for OASIS, IFS & NEMO are specified.
- Next the file called ~/config/scripts/ecearth_hres/run.sh is opened. In this script all the necessary steps for preparing a run, e.g., copying data creating namelist files based on run parameters etc are given.
- Based on the parameters provided in the above 2 steps a final runscript is automatically generated which is submitted to the scheduler.

As EC-EARTH is a multi binary run exact specification of binary – rank – node map is important for optimized run. For optimized run each component runs on sepecific group of nodes. Also due to the specific requirement of the EC-EARTH OASIS should get the 0 - (oas_numproc -1) ranks where oas_numproc is the #of OASIS procs. For bullx MPI implementation this is achieved by creating an appfile and then giving the appfile to mpirun, e.g.:

```
mpirun -f appfile.
```

A typical appfile looks like :

```
-hostfile oas_nodes -np 10 -x LD_LIBRARY_PATH -x OASIS3 -x OASIS3DEBUGLEVEL
-x LOCAL_DEFINITION_TEMPLATES  oasis3.MPI1.x
-hostfile ifs_nodes -np 512 -x LD_LIBRARY_PATH -x DR_HOOK_IGNORE_SIGNALS -x
OASIS3 -x OASIS3DEBUGLEVEL  -x LOCAL_DEFINITION_TEMPLATES  ifsmaster-
eexcon -v ecmwf -e HRES
-hostfile nem_nodes -np 320 -x LD_LIBRARY_PATH -x   opa-eexcon
```

In the above appfile example OASIS runs on 10 cpu cores, IFS runs on 512 cpu cores and NEMO runs on 320 cpu cores. OASIS, IFS and NEMO runs on nodes specified in files oas_nodes, ifs_nodes and nem_nodes respectively. The appfile and the nodelist files oas_nodes, ifs_nodes and nem_nodes is generated dynamically for every run based on the parameters specified in the step 1 above.

Scaling of High resolution EC-EARTH on Curie

The scaling of EC-EARTH run depends on choice of input parameters especially the coupling frequency, i/o frequency etc. We have chosen following parameters for our runs :

cpl freq = 3 hrs
run length = 48 hrs
ifs time step = 720 s
Frequency of history write ups = 30 ifs time steps
Frequency of spectral diagnostics = 5 time steps

nemo time step=1200 s
lim time step=3600 s
nemo frequency of write in the output file = 72 nemo time step

In a 48 hour run we have all the significant steps repeated a number of times. So although the total runtime is small it is indicative of scaling behaviour. The results of our runs are shown in Table 2 and Figure 1 and Figure 2. In figures 1 & 2 we have shown the normalized run times.

Table 2

# MPI processes				# step	average time/step (s)		
Total	NEMO	IFS	OASIS		only comp steps	only io steps	all steps
394	256	128	10	240	6.61	15.21	8.30
650	256	384	10	240	2.55	5.92	3.20
778	256	512	10	240	2.03	4.76	2.55
906	256	640	10	240	1.72	4.05	2.15
1034	256	768	10	240	1.53	3.54	1.90
1162	256	896	10	240	1.40	3.24	1.74
1290	256	1024	10	240	1.28	2.95	1.59
1418	256	1152	10	240	1.17	2.68	1.44
1546	256	1280	10	240	1.09	2.53	1.35
1610	320	1280	10	240	1.11	2.62	1.38
1866	320	1536	10	240	1.02	2.62	1.31
2634	320	2304	10	240	0.83	2.86	1.20
3402	320	3072	10	240	0.81	2.87	1.21
3482	400	3072	10	240	0.81	2.66	1.17

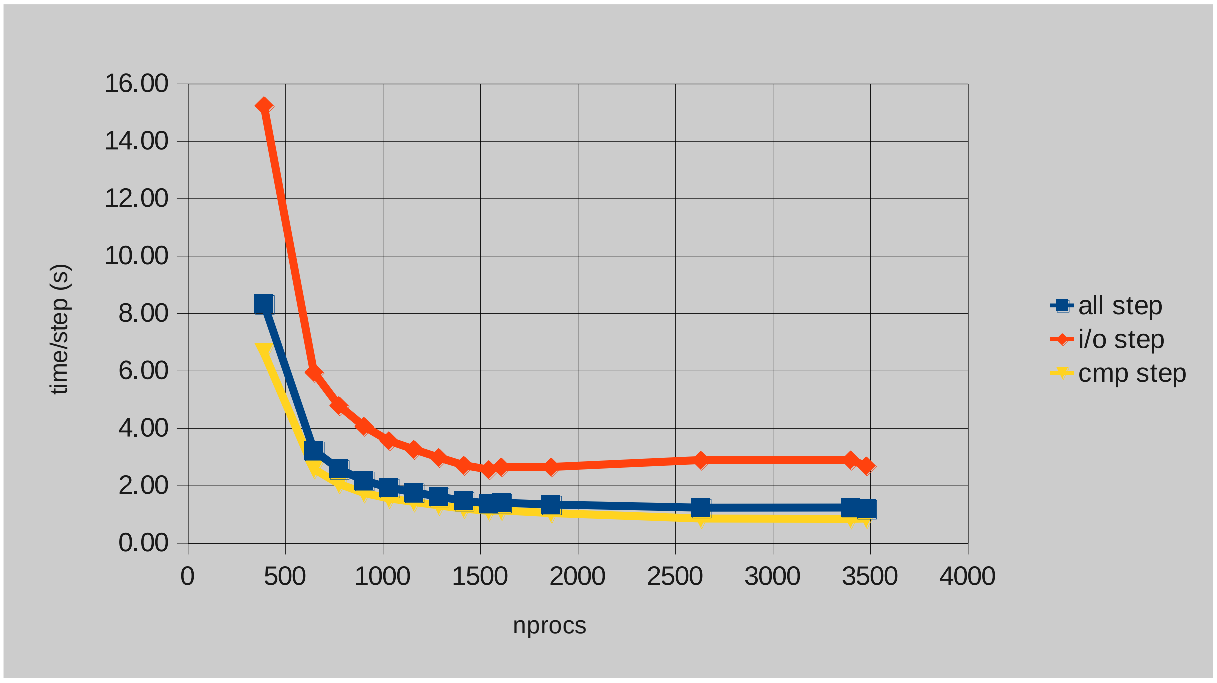


Figure 1 : Scaling of high resolution ec-earth

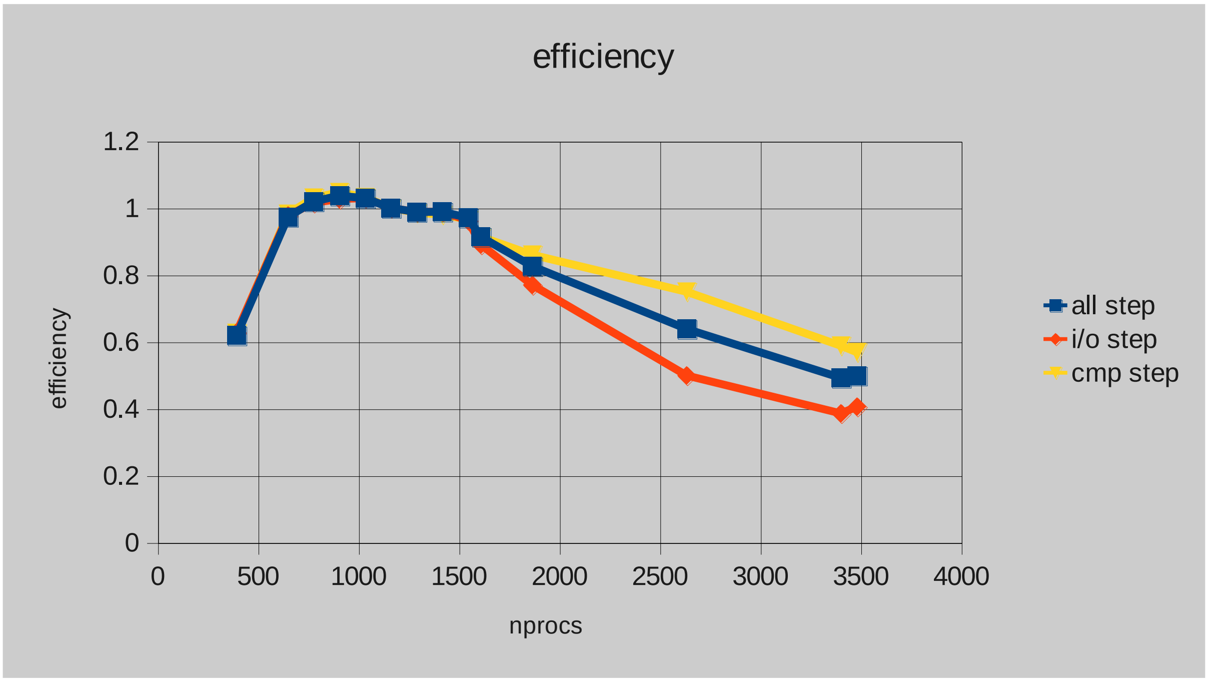


Figure 2 : efficiency of high resolution ec-earth scaling

Discussion

We have run high resolution EC-EARTH up to 3500 MPI processes. At this scale the efficiency is ~50% of the most efficient run. The most efficient run is obtained at ~1034 MPI processes. When EC-EARTH is run with less than 1034 MPI processes the efficiency is lower. This is because for less than 1034 processes NEMO processes finish earlier and wait for IFS processes.

Some time steps have only computation. Few steps additionally have i/o. These steps are significantly slower. We have shown in Table 2 the average time of only computational steps, i/o steps, and combined i/o and computational steps. This gives us an idea about which steps are bottleneck in the overall scaling. The i/o steps are 2 – 3 times more expensive than computational steps. In our experiment every 30th time step (every 6th hour) is i/o step. Actual impact of the i/o steps will depend on the i/o frequency for a particular experiment. Also the i/o file-system used for our experiment is lustre parallel file-system. Other slower file-systems can make i/o even slower. Another aspect related to i/o is some i/o steps are much more slower than other i/o steps. This is probably because of load on file-system from other programs. However on Curie the scaling of i/o steps are consistent with the computational steps. We have analyzed the file access pattern of IFS. This shows that few small files are repeatedly opened and read. These files are of few kilo-byte size and can be easily read once and kept in some variable for later accesses. This could improve the file i/o considerably. However as IFS is a complex code we have not tried to do the code change ourselves and would expect that developers of the code would implement such features in future versions of the IFS code.

Hybrid MPI-OpenMP EC-EARTH run

The coupled EC-EARTH code is generally run in MPI only mode. The coupled EC-EARTH has three components namely IFS, NEMO and OASIS respectively for atmosphere, ocean and coupling. While NEMO and OASIS codes are MPI only, IFS code has built in support for OpenMP along with MPI. Standalone IFS can be run in hybrid OmpenMP + MPI mode. There can be various advantages of hybrid runs compared to MPI only runs, e.g., less memory consumption, better scaling etc. This can be particularly beneficial for very wide runs. However it was not clear whether in coupled setting IFS can be run in hybrid mode. In the last part of our ISENES – JRA2 work we have tried successfully to run coupled EC-EARTH in hybrid OpenMP – MPI mode.

Compilation and running

For our tests we took the recent version of EC-EARTH, revision no. 1117. The compilation of the code was done following standard EC-EARTH compilation procedure. We use Intel compiler v-12.1.4 and Intel MPI v-4.0.3.008. We add extra compiler / linker flag “-openmp” for the OpenMP compilation. For OASIS and NEMO “-openmp” flag has no impact while for IFS it is compiled with OpenMP.
Running of the coupled EC-EARTH with one part (IFS) in hybrid mode is bit challenging. For a normal MPI program is easy to set number of threads for the entire run to a desired value by setting the environment variable OMP_NUM_THREADS. A typical example for this is:

```
export OMP_NUM_THRAEDS=2
mpirun -np 2 exe1: -np 2 exe2: -np 2 exe3
```

In this example of multi-binary run each of exe1, exe2 and exe3 will run on two MPI ranks each, with each rank spawning two threads. This will not work for EC-EARTH. So we have “embedded” export OMP_NUM_THRAEDS=value inside ifs. To do this we move IFS binary called ifs to something called ifs.orig. Then we create an executable shell script called ifs which contains:

```
if [ -n "$IFS_NUM_THREADS" ]; then
  export OMP_NUM_THREADS="$IFS_NUM_THREADS"
  export KMP_STACKSIZE=500m
  eval ifs.orig "$@"
fi
```

As can be seen above the ifs script sets the OMP_NUM_THREADS and KMP_STACKSIZE and then calls the original ifs binary. With this setup the EC-EARTH invocation looks like

```
export IFS_NUM_THREADS=2
export OMP_NUM_THREADS=1
mpirun -n n1 oasis : -n n2 ifs -v ecmwf -e ECE3 : -n n3 nemo
```

As the IFS_NUM_THREADS is set to 2 the IFS binary, ifs.orig will spawn two threads per rank while nemo and oasis binaries will be single threaded.

Results and discussions

For our experiments we take NEMO ORCA1L46_LIM3 configuration, IFS grid T255L91, Coupling frequency 3 hrs, IFS time step 2700s, NEMO time step 3600 s, LIM time step 3600s. Total run length is 30 days. The results are shown in table 1. In column 3 of the Table 1 we show the average IFS step time excluding the coupling and i/o steps. As we can see IFS step times are slightly less for threaded runs. On the other hand the memory consumption is nearly 40% less in the threaded run. Total run time is 15% faster for threaded run. This impact may be more prominent for wider runs. In ISENES2 we would like to pursue this further.

Table 1

Coupled EC_EARTH run with NEMO 64 ranks, OASIS 10 ranks

Type of run	Totals IFS memory (GB)	IFS time step (s)	Total run time (mm:ss)
128 IFS ranks, 1 thread each	110	1.02	22:17
64 IFS ranks, 2 threads each	76	0.9	19:27

EC-CONF: A New Approach for Build and Run Configuration of EC-Earth

Contributors: Uwe Fladrich, SMHI

Software configuration can be a cumbersome task, particularly when it comes to complex systems with many components, different processing stages, and numerous files to be considered. Whenever more than a few configuration parameters, files and syntaxes are involved, or as soon as the configuration is to be done frequently, the process is error prone, too. Consistency and reproducibility are not easily achieved and documentation of a certain configuration is usually not included in the process.

The ec-conf tool is developed to aid the process of software configuration and to mitigate the problems just mentioned. As such, ec-conf is designed for the integration into the building and running cycle of a complex software system. Although ec-conf is being developed in the context of the coupled climate model EC-EarthÂ 3, it is not, per se, dependent on this software.

Configuration is needed, for example, when the software is built (i.e. compiled and linked) or when it is run in the context of a certain computing experiment. The configuration has to be redone whenever relevant aspects â€“ such as the computing platform, user environment, or experiment settings â€“ change.

The fundamental principle behind ec-conf is that the process of software configuration implies the modification of configuration files according to configuration parameters. Configuration files are assumed to be text files and configuration parameters comprise a name and a value. The configuration parameter value is to be adapted according to the situation that the software is to be configured for.

That is, ec-conf creates configuration files by modifying configuration parameters according to their required values. This is achieved by reading the configuration parameters from an XML data base file and creating the configuration files with the help of templates. In order to control this process, ec-conf provides both a command line interface and a graphical user interface (GUI). Both interfaces can be used interchangeably, although they provide slightly different levels of control and comfort.

The main advantage of using ec-conf lies in the â€œone place, one syntaxâ€• principle, which implies that configuration is confined to just one file, where all configuration parameters are condensed, and, consequently, to one syntax. This principle holds regardless of the number of components and the number of configuration files and file types that are part of the software system.

Ec-conf has been implemented in Python, which should ensure widespread portability. Care has been taken to minimise the number of Python modules needed, although programming convenience has not been overly compromised. The ec-conf GUI makes use of Tkinter, however, the command line interface can be run even when Tkinter is not available.

In order to utilise ec-conf, a user has to provide an XML data base file that contains all configuration parameters and a number of template files that are used to create the configuration files. These files are usually created only once upon ec-conf's first use. Later on, only limited modifications are needed.

It is possible (and recommended) to introduce ec-conf gradually, which eases the transition from manual configuration. This is done by selecting just one configuration file as a template and start with few configuration parameters. Hence, only two files (one XML data base and one template) have to be provided for a start. In fact, any configuration file can be used as ec-conf template file without modification. It means that ec-conf is just passing the content of the template file without any changes. However, this implies that ec-conf has to be complemented by manual configuration.

When both the XML data base file and template file(s) are in place, the regular usage of ec-conf consists of two steps:

- Adapting the configuration parameter values in the XML data base file, and
- Running ec-conf in order to create the configuration files.

The first step is accomplished by editing the XML data base file (which is a text file) with a text editor of the user's choice. The second step means running either the command line or graphical user interface (GUI) of ec-conf. Alternatively, both steps can be performed in one go with the graphical user interface. The GUI can be used instead of the command line interface in order to use all functions of ec-conf, in fact, the GUI is preferable for a more interactive way of working with ec-conf.

NEMO Porting, benchmarking and optimization on Marenostrum

Contributors: I.Epicoco, S.Mocavero, G.Aloisio
Euro-Mediterranean Center for Climate Change (CMCC)

Introduction

NEMO (Nucleus for European Modeling of the Ocean) is a 3-dimensional ocean model used for oceanography, climate modelling as well as operational ocean forecasting. It includes other sub-component models describing sea-ice and biogeochemistry. Many processes are parameterized, e.g. convection and turbulence. It is used by hundreds of institutes all over the world. The open-source code consists of 100k lines of code, it is developed in France and UK by the NEMO development team and it is fully written in Fortran 90. The MPI paradigm is used to parallelize the code. NEMO is based on a finite-difference model with a regular domain decomposition and a tripolar grid to prevent singularities. It calculates the incompressible Navier-Stokes equations on a rotating sphere. The prognostic variables are the three-dimensional velocity, temperature and salinity and the surface height. To further simplify the equations it uses the Boussinesq and hydrostatic approximations, which e.g. remove convection. It can use a linear or non- linear equation of state. The top of the ocean is implemented as a free surface, which requires the solution of an elliptic equation. For this purpose, it uses either a successive over-relaxation or a preconditioned conjugate gradient method. Both methods require the calculation of global variables, which incurs a lot of communications (both global and with its nearest neighbours) when multiple processors are used. The scientific literature reports several performance analyses of NEMO model, using different configurations and with several spatial and time resolutions. At the National Supercomputer Centre (NSC) the porting, optimization, tuning, scalability test and profiling of NEMO model on linux - X86-64 - infiniband clusters, have been performed. ORCA1 configuration available from NOCS website has been used for scaling/benchmark studies. Within the PRACE project, a benchmark activity report on several applications has been produced. The NEMO code has been ported and evaluated on several architectures such as the IBM Power6 at SARA, the CRAY-XT4, the IBM BlueGene [1].

Analysis of Scalability

The analysis of scalability of the parallel code aims at verifying how much it is possible to increase the complexity of the problem, in terms of spatial and time resolution, scaling up the number of processes. As first step of the analysis we profiled the original NEMO code for highlighting possible bottlenecks slowing down the efficiency, when the number of processes increases.

Model Configuration

The NEMO configuration taken into account is based on the official release (v3.2) with some relevant improvements introduced by INGV (Istituto Nazionale di Geofisica e Vulcanologia - Italy). Moreover it is tailored on the Mediterranean Basin. The Mediterranean Sea is both too complex and too small to be adequately resolved in global-scale climate and ocean-only models. To properly address some key processes, it is necessary to adequately represent the general circulation of the Mediterranean basin, the fine-scale processes that control it (e.g. eddies and deep convection), and the highly variable atmospheric forcing. A high-resolution general circulation model of the Mediterranean Sea has been developed in the last 10 years to provide operational forecast of the ocean state [2]: the Mediterranean ocean Forecasting

System. The physical model is currently based on version v3.2 of NEMO and is configured on a regular grid over the Mediterranean basin plus a closed Atlantic Box. Horizontal resolution is 1/16 x 1/16 degrees with 72 vertical Z-levels. The model is forced with meteorological data that are either read from gridded external datasets or interpolated on line to the model grid. The model salinity and temperature fields along the boundary of the Atlantic box are relaxed at all depth to external data (open boundary conditions [3]). This is done within an area which has an extension of 2° at the west and south boundary and 3° at the northern boundary (in order to cover all the area of the Gulf of Biscay). This configuration, even more if coupled with biochemical models, poses several computational challenges:

- High spatial resolution with many grid points and a small numerical time step
- Presence of open boundaries, which implies that additional data need to be read by selected sub-domains
- Computation of diagnostic output across sub-domains
- Storage of large amount of data with various time frequencies.

Profiling

The research activity has been carried out at the Barcelona Supercomputing Center using the MareNostrum cluster. It is one of the most powerful systems within the HPC-Ecosystems in Europe. It has a calculation capacity of 94.21 Teraflops. One of the key issues that characterize MareNostrum is its orientation to be a general purpose HPC system. The computing racks have a total of 10240 processors. Each computing node has 2 processors PowerPC 970MP dual core at 2.3 GHz, 8 GB of shared memory 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. The default compilers installed are IBM XL C/C++, and IBM XL FORTRAN. There are also available the GNU C and FORTRAN compilers. The MareNostrum uses GPFS as high-performance shared disk file system that can provide fast, reliable data access from all nodes of the cluster to a global file system. Moreover, every node has a local hard drive that can be used as a local scratch space to store temporary files during the execution of user’s jobs. All data stored in these local hard drives will not be available from the login nodes. The first evaluation was focused on establishing how much the computational performance are influenced using the GPFS file system or the local disks. The results, showed in figure 1 and analytically reported in table 1, highlight that the exploitation of local disks can reduce the wall clock time up to 40% against using the GPFS file system. The performances of the GPFS are strictly related to the actual load of the whole cluster and hence they are very variable during the time. Since the local disks are not accessible from the login node, some modifications to the NEMO runscript file, used to launch the model, have been performed.

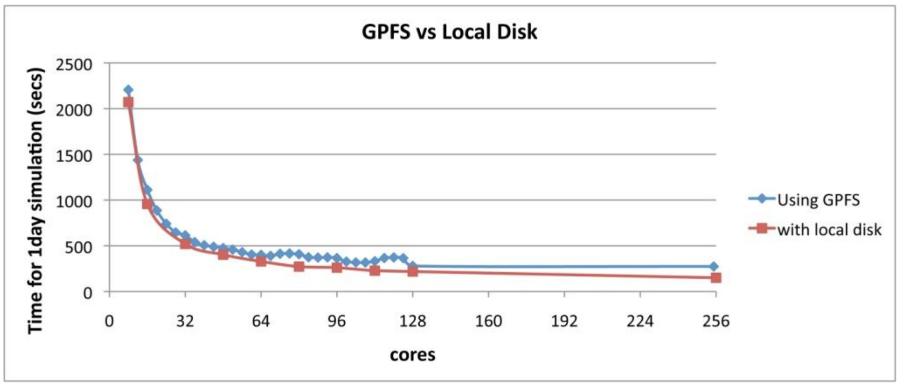


Fig. 1 - Execution time on 1day simulation: GPFS vs local disk

Cores	GPFS	Local Disk	Speedup
8	2205.86	2071.45	1.065
16	1112.19	957.17	1.162
32	614.02	519.91	1.181
48	474.65	402.41	1.18
64	401.29	328.58	1.221
80	407.67	272.55	1.496
96	365.64	260.42	1.404
112	331.35	227.85	1.454
128	279.75	218.35	1.281

Table 1 - Execution time on 1day simulation: GPFS vs local disk

The NEMO code supports 2D domain decomposition. The size and the shape of the sub domain assigned to each parallel process impact on the overall performance. The second step of performance analysis has been focused on the impact of the domain decomposition on the wall clock time. The analysis of the scalability has been performed taking into account a 1D decomposition (both horizontal and vertical) and a 2D decomposition. The 2D decomposition has been chosen such that the local sub domain would have a square shape. The experimental results demonstrate that the best performance is achieved using a 2D decomposition as showed in figure 2.

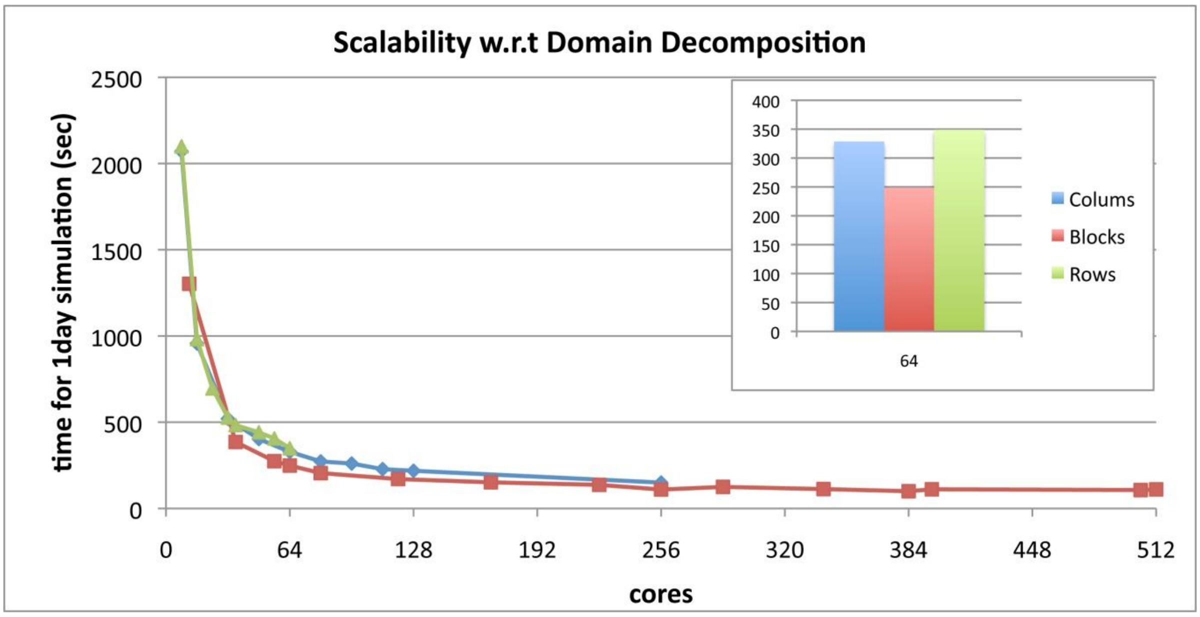


Fig. 2 - Execution time on 1day simulation w.r.t. domain decomposition.

The overall evaluation of the legacy code has been carried out in order to analyze the parallel behaviour of the application. In particular at high level we have taken into account two metrics: the parallel scalability and the parallel efficiency. Both metrics provide an overall evaluation on how much the code is well parallelized. These measures can guide further analysis focused on specific aspects of the code. It is worth noting here that the approach followed for the analysis started considering the application as a black box. The complexity of the code makes quite unfeasible the definition of a reliable theoretical performance model. The approach we followed was based on an experimental approach. The parallel efficiency and speed-up, respectively reported in table 2 and figure 3, have been evaluated taking as reference time the wall clock time of the application with 12 processes. Due to the amount (8 GB) of main memory per node available on MareNostrum, the execution of the sequential version of the model is prohibitive requiring at least 20 GB with this configuration. The experimental results showed a limit of the scalability up to about 192 cores.

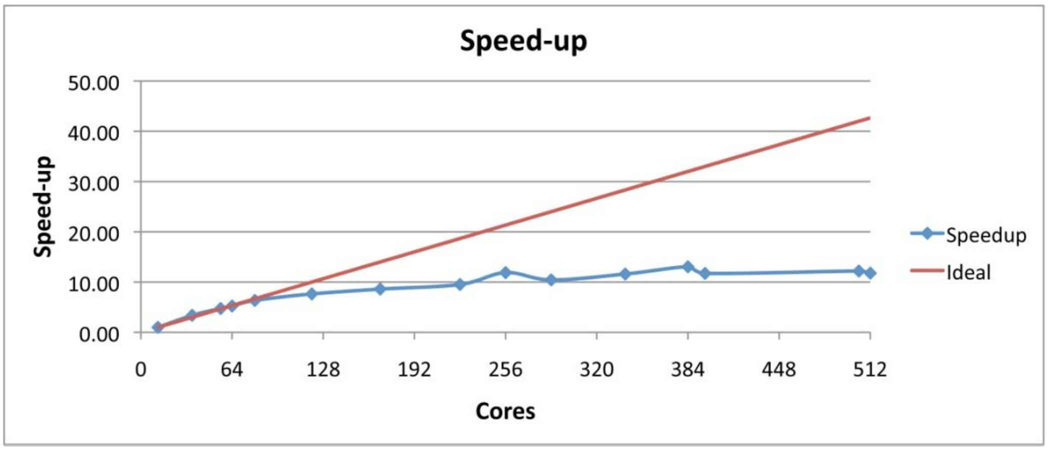


Fig. 3 - Speed-up of original version. Speed up is relative to the performance of that on 12 processes (due to memory requirements).

Decomposition	Cores	1day sim (s)	Eff. (%)	SYPD	1Y sim (h)

6x2	12	1302.54	100	0.18	132.06
16x4	64	248.71	98.2	0.95	25.22
20x6	120	170.37	76.46	1.39	17.27
128x2	256	109.57	55.72	2.16	11.11
32x9	288	124.84	43.47	1.9	12.66
34x10	340	112.28	40.94	2.11	11.38
128x3	384	99.87	40.76	2.37	10.13
36x11	396	111.08	35.53	2.13	11.26
128x4	512	110.57	27.61	2.14	11.21

Table 2 - Original code performance. Efficiency is relative to the performance of that on 12 processes (due to memory requirements).

In order to perform a deeper investigation on the motivation for the poor efficiency, we have analyzed the scalability of each routine in the code. For identifying those routines with a relevant computational time we used the gprof utility and the Paraver [4] tool with dynamic instrumentation of the code. Figure 4 shows a paraver snapshot of a very short run (just 3 time steps). Different colours represent different states of the run: blue for computation, red for I/O operations, orange and pink respectively for global and point-to-point communications. *opa_init* initializes the parallel environment and synchronizes processes. Its execution time is negligible when the number of steps is high. The first and the last time steps perform some IO operation, respectively reading input and restart and writing output and restart.

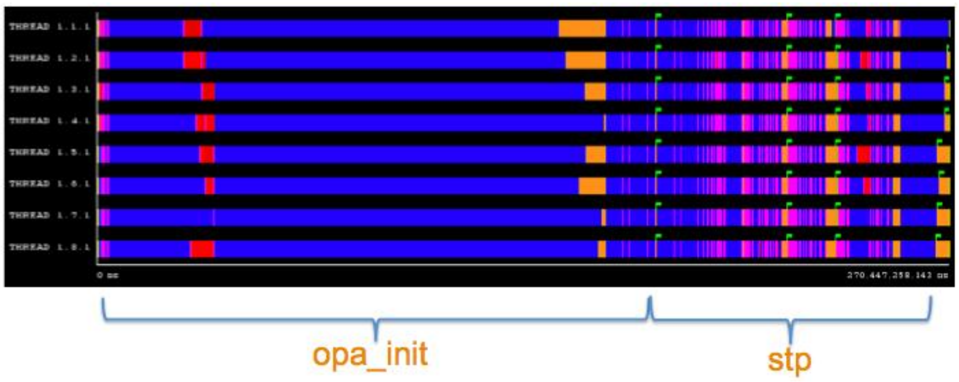
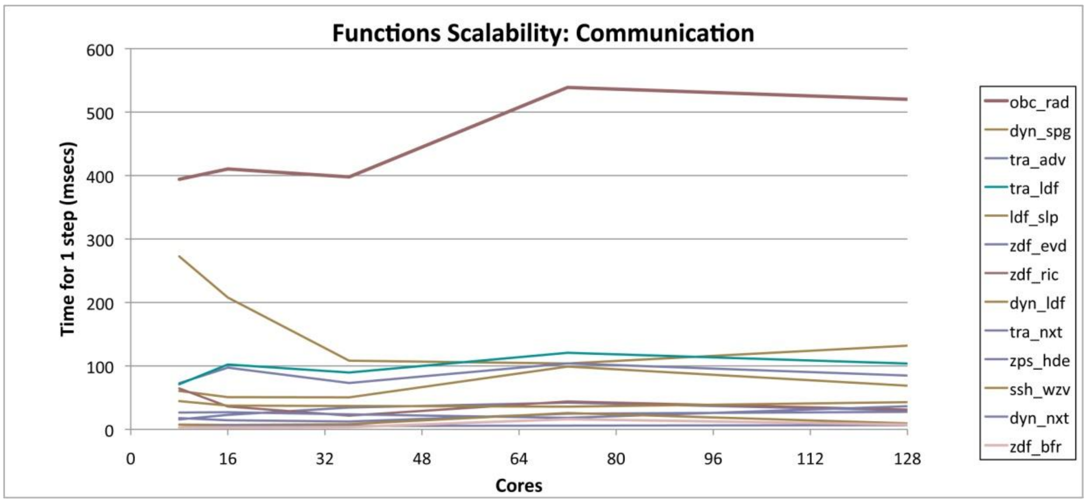
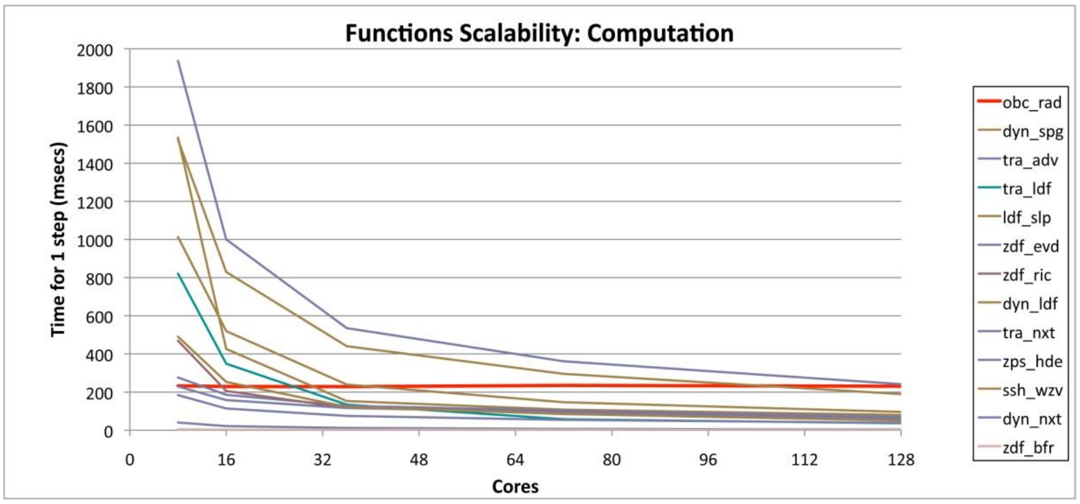


Fig 4 - Paraver trace for a NEMO run.

The analysis has been restricted to a single time step: we chose a "general" time step, considering as "general" those time steps with operations occurring every time. Indeed some "occasional" operations like reading the open boundaries values, or storing the state variable values, occur only for some particular time step. We have identified about 36 routine of interest and we have evaluated their scalability running the application with 8, 16, 36, 72 and 128 processes. For each routine we have taken into consideration both computing and communication time. The results of the analysis are reported in figure 5.





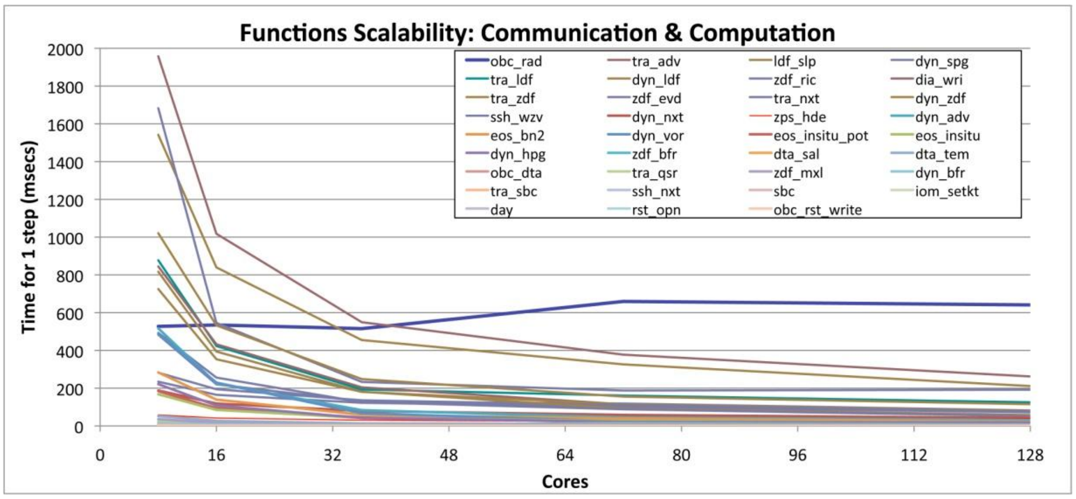


Fig. 5 - NEMO functions scalability: computation, communication & total time.

The analysis immediately highlighted the *obc_rad* routine (in charge of calculate the radiative velocity on the open boundaries), the *dyn_spg* (in charge to solve the elliptic equation for the barotropic function) and the *tra_adv* (in charge to evaluate the advection transport of the fields), as those routines to be deeper investigated.

Optimization

The optimization phase aimed at redesign critical part of the NEMO code taking into account the following main aspects:

- Exploitation of the memory hierarchy. A relevant limitation of the performance is strictly related to redundant memory accesses or to a high level of cache miss ratio
- The I/O operations are one of the critical factors that limit the performance and the scalability of a climate model. The I/O pattern implemented in NEMO can be classified as: read once and write periodically
- The communication among parallel processes plays a crucial role on the performance of a parallel application. Several good practices can be followed in order to reduce the communication overhead, such as modifying the communication pattern in order to overlap communication and computation; joining several short messages sent with several MPI calls in a bigger one sent once.

The analysis of the scalability showed a limit at 192 cores due to a high level of communication overhead. The bottleneck has been identified in the function responsible for the evaluation of the open boundaries conditions. After the evaluation of the open boundaries, the processes exchange the overlapped values over the boundaries with their neighbours. The function take more than 60% of its time in communication.

OBC_RAD Routine

As already stated before, the NEMO configuration we used for our analysis is limited to an oceanic region and namely the Mediterranean basin, which communicates with the rest of the global ocean through "open boundaries". An open boundary is a computational border where the aim of the calculations is to allow the perturbations generated inside the computational domain to leave it without deterioration of the inner model solution. However, an open boundary has also to let information from the outer ocean enter the model and should support inflow and outflow conditions. The open boundary package OBC is the first open boundary option developed in NEMO. It allows the user to:

- Tell the model that a boundary is "open" and not closed by a wall, for example by modifying the calculation of the divergence of velocity there
- Impose values of tracers and velocities at that boundary (values which may be taken from a climatology): this is the "fixed OBC" option
- Calculate boundary values by a sophisticated algorithm combining radiation and relaxation ("radiative OBC" option).

The Open Boundaries calculation is performed within the *obc_rad* routine. The current implementation of the *obc_rad* function swaps arrays to calculate radiative phase speeds at the open boundaries and calculates those phase speeds if the open boundaries are not fixed. In case of fixed open boundaries the procedure does nothing. In particular the following algorithmic steps are performed: (i) each MPI process calculates the radiative velocities on its subdomain starting with zonal velocity field; (ii) the data on the border of the local sub-domain are exchanged among MPI processes with a cross communication pattern; (iii) repeat from step one for the following fields: tangential velocity, temperature and salinity. In the worst case, when the whole domain has 4 open boundaries (east, west, north and south) each MPI process performs 16 exchanges (4 fields exchanges multiplied by 4 open boundaries). For each field, an MPI process sends and receives the data to/from 4 neighbours. Even though the exchanged fields are 3D arrays, the current implementation of the communication routine (named *mppobc*) calls iteratively a library routine for sending/receiving 2D arrays. Figure 6 shows the original communication pattern among processes.

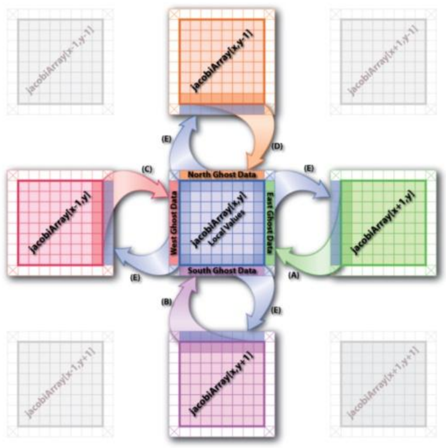


Fig. 6 - Communication pattern during the open boundaries evaluation: before optimization all processes were involved in the communication.

OBC_RAD Optimization

The analysis of the scalability showed that the communication overhead within the *obc_rad* function reaches a ratio of 74% running the model with 8 cores. The main limits to the scalability have been then identified in a heavy use of communication among processes. With a deeper analysis of the *obc_rad* algorithm we noticed that several calls to the MPI send/MPI recv were redundant and hence they could have been removed. Figure 7 illustrates the essential communications needed for exchanging the useful data on the boundaries.

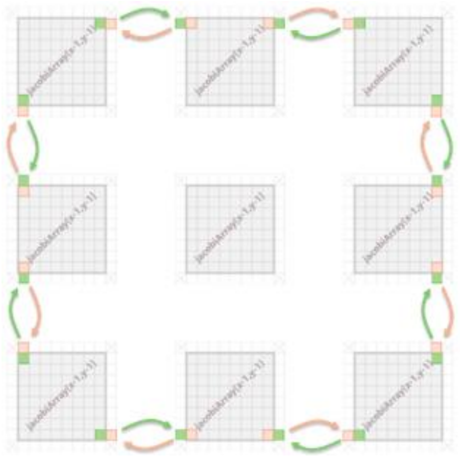


Fig. 7 - Communication pattern during the open boundaries evaluation: after optimization only the processes on the boundaries were involved in communication and they exchange only the data on the boundary.

The optimization reduced the communication time through the following actions: the processes on the borders are the only processes involved in the communication; the data exchanged between neighbours are only the data on the boundary; the data along the vertical levels are "packed" and sent with only one communication invocation. Figure 8 shows how communications (yellow lines) within the *obc_rad* are drastically reduced after the

optimization.
The analysis of the scalability of NEMO using the optimized version of the *obc_rad* routine has been performed starting from a configuration on 12 cores with a decomposition 6x2 up to 512 cores (128x4) on 1-day simulation. As reported in table 3, the minimum wallclock time happens on 396 cores with a decomposition 36x11. Efficiency increases compared with the original version, as well as the parallel speedup (figure 9), and the *obc_rad* execution time was reduced of about 33.81%.

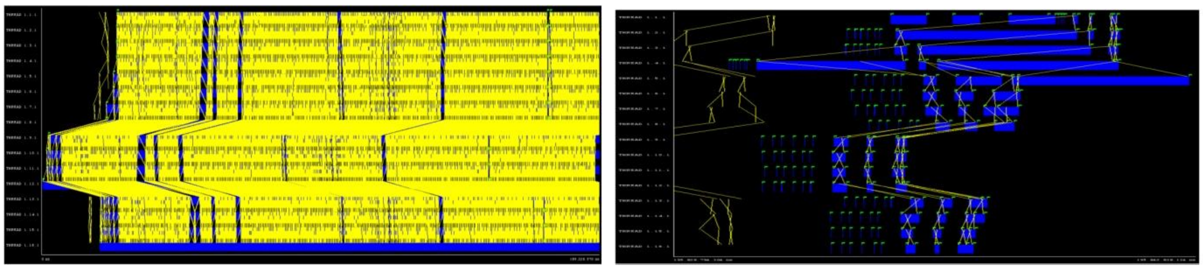


Fig. 8 - Communications within OBCRAD before and after the optimization.

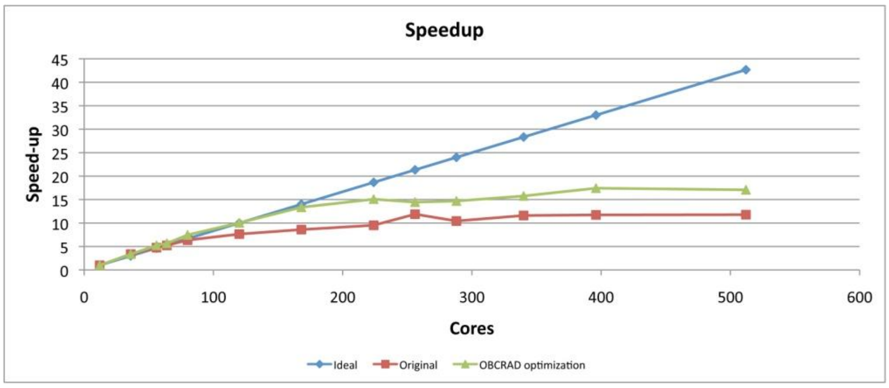


Fig 9 - Speed-up: OBCRAD optimized vs original version.

Decomp.	Cores	Original exec. time (s)	Original efficiency (%)	OBCRAD optimized exec. time (s)	OBCRAD optimized efficiency (%)
6x2	12	1302.54	100	1281.28	100
12x3	36	385.32	112.68	382.47	111.67
14x4	56	274.22	101.79	244.73	112.19
16x4	64	248.71	98.2	226.17	106.22
16x5	80	205	95.31	171.54	112.04
20x6	120	170.37	76.46	127.54	100.46
24x7	168	151.45	61.43	95.98	95.35
28x8	224	136.78	51.02	84.95	80.8
128x2	256	109.57	55.72	88.7	67.71
32x9	288	124.84	43.47	87.24	61.2

34x10	340	112.28	40.94	81.26	55.65
36x11	396	111.08	35.53	73.53	52.8
128x4	512	110.57	27.61	75.02	40.03

Table 3 - OBCRAD optimized vs original version: performance analysis.

SOL_SOR Routine

After the *obc_rad* optimization, a new detailed analysis of scalability on all of the above mentioned 36 functions has been performed. It allowed identifying the SOR solver routine (called by the *dyn_spg* function) as the most expensive from the communication point of view. The function implements the Red-Black Successive-Over-Relaxation method [5], an iterative search algorithm used for solving the elliptical equation for the barotropic stream function. The algorithm iterates until convergence for a maximum number of iterations. The high frequency of exchanging data within this function increases the total number of communications.
At each iteration, the generic process computes the black points inside the area, updates the black points on the local boundaries exchanging values with neighbours, computes red points inside and finally updates red points on the local boundaries (always exchanging with neighbours). Each process exchanges data with 4 (at north, south, east and west) of its 8 neighbours: the order of data transfer guarantees data reliability. Communications are very frequent and the total number of exchanges is given by the number of iteration multiplied by 2 (one for red points, and one for black) by 4 neighbours. The *sol_sor* function, implementing the SOR solver method, calls the *lbc_lnk_2d* e function for exchanging data among processes. Both the functions are characterized by two components: a running component respectively computing and buffering data before sending and after receiving and a communication one (within the *sol_sor* there is a group communication during the convergence test).

SOL_SOR Optimization

The algorithm of *sol_sor* suggests a possibility to improve performance, especially when the number of processes increases. At each iteration, communication and computation could be overlapped. The algorithm can be modified as follows: (i) computing of data on the local boundaries, (ii) communication of computed region overlapped with computation of the inner domain. This solution has been implemented, but it did not give expected results. The original version uses blocking communications during the exchange. The modified version was implemented by the use of non-blocking communications to allow message transfer to be overlapped with computation. As result, not only running, but also communication time was increased after the code modification. Changing communication algorithm, the computation within *sol_sor* has been split in two steps. This generated an access to non-contiguous memory locations with a consequent increase of L1 cache misses. More cache misses means more instructions and then more computing time. Moreover, the introduction of non-blocking communication does not guarantee the order of data exchanging among processes, so that a generic process needs to communicate not only with the north, south, east and west processes, but also with the diagonal ones, doubling the number of communications. Since communication and computation are overlapped, the increase of communications number should not increase the execution time. However, the behaviour of communications on MareNostrum has not been the expected one. Using the Dimemas [6] tool, we have theoretically evaluated the new algorithm on an ideal architecture with the nominal values declared for MareNostrum (figure 10). Even though from a theoretical point of view the new algorithm performed better than the old one, the experimental results did not confirm the expectations. One of the possible could be the implementation of the non-blocking communication within the installed MPI library. Moreover it is worth noting here that with a high level of parallelism, the *sol_sor* function has a fine computational granularity, so that the execution time feels the effects of several causes not directly related to the application but also to the system or to the tracking tools and it is very difficult to estimate its behaviour. In these cases the only thing is to consider the experimental results, which on MareNostrum highlight better performances of the original version.

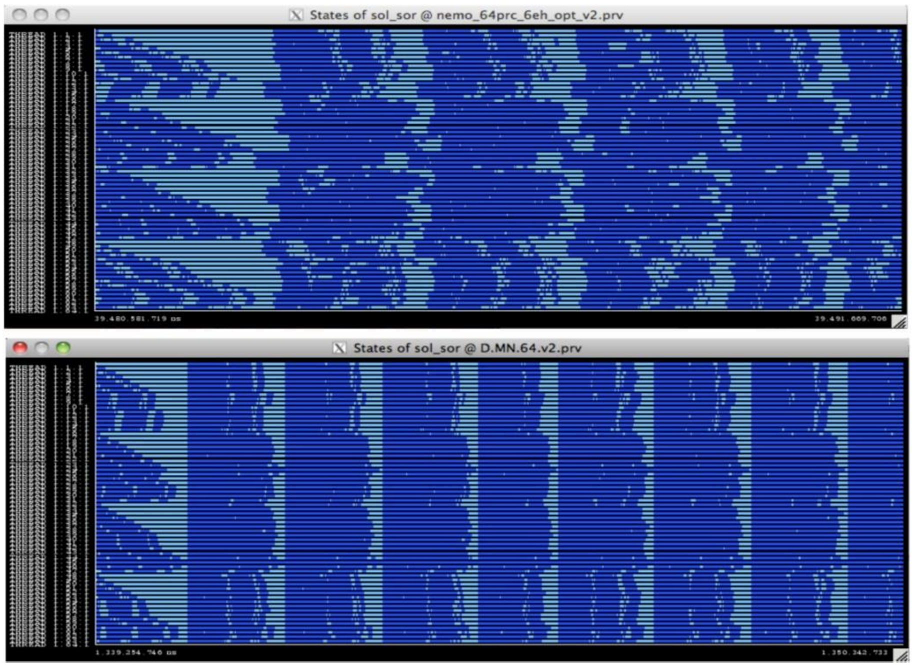
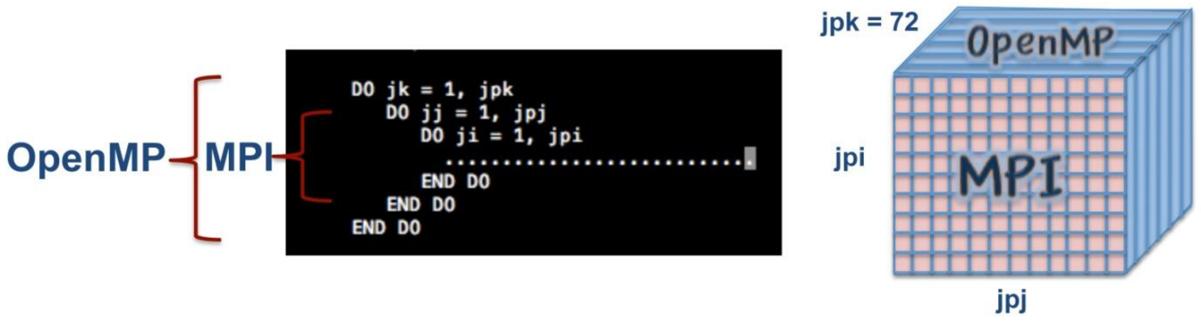

States of sol_sor @ nemo_64prc_6eh_opt_v2.prv
States of sol_sor @ D.MN.64.v2.prv

Fig. 10 - Analysis by Dimemas: (a) real behavior of communications on MareNostrum, (b) expected behavior of communications on MareNostrum.

Parallelization Improvement

Many NEMO routines are characterized by operations performed on a 3D domain, along jpi, jpj and jpk as showed in figure 11. The MPI parallelization exploits the domain decomposition on 2 dimensions (along jpi and jpj). In order to reduce the computational time, a hybrid parallel approach could be introduced. An additional level of parallelization, using the OpenMP shared-memory paradigm, could work on vertical levels, which are fixed for our NEMO configuration to 72.



```
DO jk = 1, jpk
   DO jj = 1, jpj
      DO ji = 1, jpi
         ..........................
      END DO
   END DO
END DO
```

Fig 11 - OpenMP parallelization applied to 3D domain decomposition.

Before modifying the code, an estimation of the percentage of the application, which should benefit from the use of OpenMP is needed. Using the gprof utility the percentage of time spent by functions called by the step routine (simulating a time step) and containing loops on levels without dependences, has been computed. It was about the 83% of the total computational time. OpenMP parallelization has been introduced within all of these functions. Fixing the number of allocated cores, we can execute the application using only MPI (in this case the number of MPI processes will be equal to the allocated cores) or using MPI/OpenMP parallelization (in this case, in order to better exploit the MareNostrum architecture, we created 4 OpenMP thread for each MPI process). A Paraver analysis of the duration of the functions called in the main loop over the time steps has been performed. Functions execution is more balanced among threads using the hybrid version due to the reduced number of communications (figure 12 shows how time spent waiting for communication, the white lines, has been reduced).

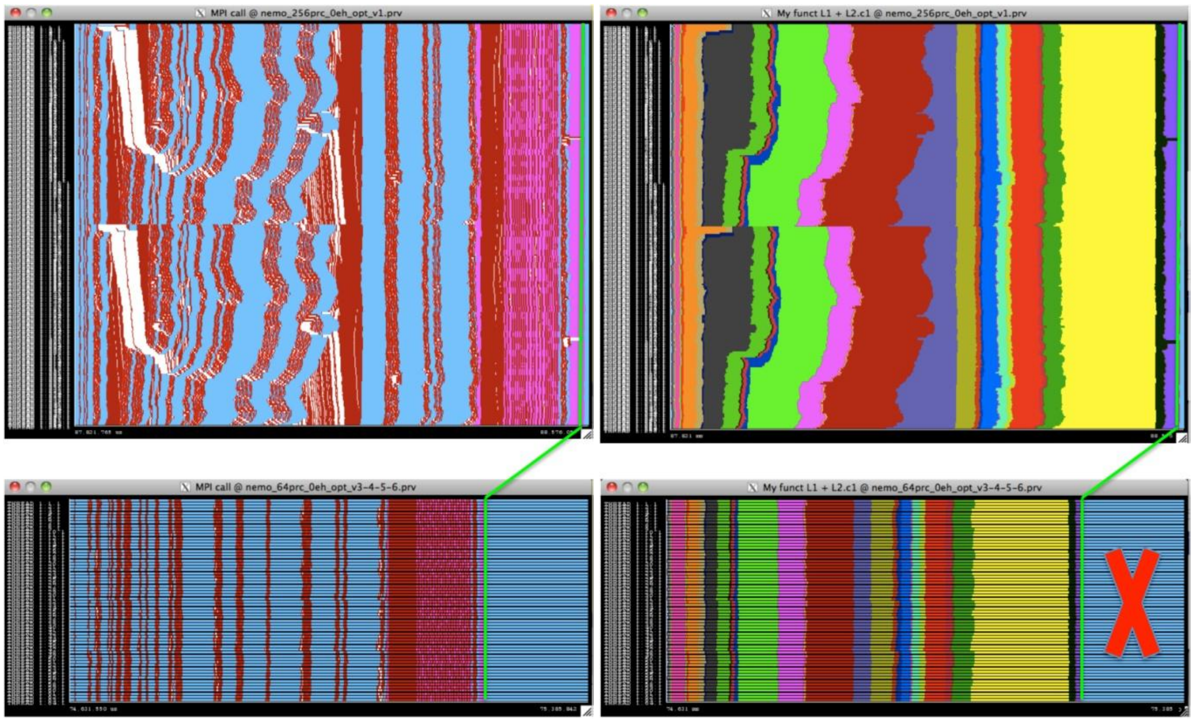

MPI call @ nemo_256prc_0eh_opt_v1.prv
My funct L1 + L2.c1 @ nemo_256prc_0eh_opt_v1.prv
MPI call @ nemo_64prc_0eh_opt_v3-4-5-6.prv
My funct L1 + L2.c1 @ nemo_64prc_0eh_opt_v3-4-5-6.prv

Fig 12 - OpenMP parallelization: (a) communications and (b) routines execution time using 256 cores for 256 MPI procs, and using 256 cores for 64 MPI procs, each one with 4 threads

With the OpenMP parallelization, the parallel speed-up improved, as shown in figure 13. The benefits derived from the hybrid parallelization can be appreciated when the number of MPI processes exceeds 30 and consequently the total threads number exceeds 120.
Table 4 shows performance results in terms of execution time and efficiency, on 1-day simulation, comparing the original code with the version after the optimization of the *obc_rad* routine and after the introducing of the second level of parallelism. The minimum wallclock time happens on 396 cores. Efficiency increases compared with both the original version and the *obc_rad* optimized one; execution time is reduced of about 18%.

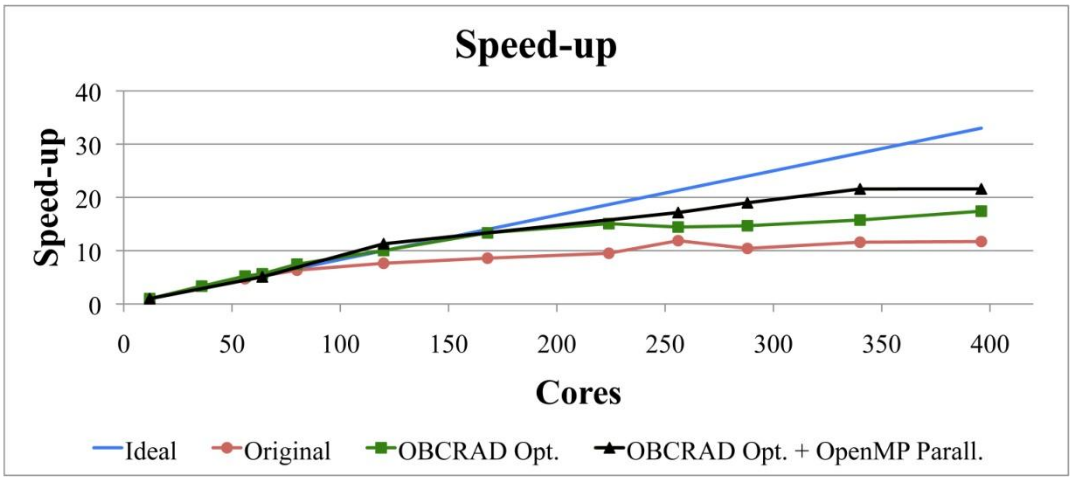


Fig. 13 - Speed-up: OpenMP parallelized vs previous versions.

MPI Decomp.	Cores	Original exec. time (s)	Original efficiency (%)	OBCRAD optimized exec. time (s)	OBCRAD optimized efficiency (%)	OpenMP exec. time (s)	OpenMP efficiency (%)
3x1	12	1302.54	100	1281.28	100	1767.3	100
8x2	64	248.71	98.2	226.17	106.22	346.75	95.57
10x3	120	170.37	76.46	127.54	100.46	156.48	112.94
16x4	256	109.57	55.72	88.7	67.71	75.89	80.46
18x4	288	124.84	43.47	87.24	61.2	68.57	79.15
17x5	340	112.28	40.94	81.26	55.65	60.34	76.19
33x3	396	111.08	35.53	73.53	52.8	60.29	65.47

Table 4 - OpenMP parallelized vs previous versions: performance analysis.

PELAGOS025

As further development, a new configuration of NEMO has been analyzed. We analyzed the parallel scalability of the NEMO global oceanic model with a configuration of 0.25 degree, coupled with the BFM biogeochemical flux model (Vichi et al., 2007 - http://bfm.cmcc.it). The oceanic model has an horizontal resolution of 0.25 degree, that corresponds to a 1442x1021 grid points, and 50 vertical levels; the biogeochemical model uses the same resolution and grid of the oceanic model and it takes in consideration 57 state variables (tracers).

An analysis of scalability of the coupled model has been performed on the Calypso parallel cluster architecture at the CMCC (Euromediterranean Center on Climate Change, Italy) Supercomputing Center. Each node of the cluster is equipped with 16 Power6 CPUs dual core, for a total of 32 cores and 128GB of total memory per node (4GB per core). The architecture supports the Simultaneous Multi Threading, which allow the parallel execution of two threads on the same core. The NEMO model, characterized by an intensive use of the memory, is not able to exploit SMT, so 32 cores per node have been used.

Due to the memory requirements, the model ORCA025-BFM has been executed on a minimum of 5 nodes up to the maximum available nodes (21).

The execution time is strictly related to the balance of the ocean points among the parallel processes. Fixed the number of processes, we used the *mpp_opt* tools (available within the NEMO package) to find the best domain decomposition excluding those subdomains with only land points. The optimal decomposition has to be defined in order to optimize the load balance. The table 5 reports the domain decomposition used for each parallel configuration. The Balance column reports the ratio between the average number of ocean points and the maximum number of ocean points. A perfect balance is given when the ratio is 1.

MPI Decomp. (jpni x jpnj)	Cores (jpnij)	Number of nodes	Num of Ocean Points for the heaviest process	Balance
6 x 29	160	5	413020	0.5273
6 x 35	192	6	345022	0.5319
50 x 5	224	7	293849	0.5361
8 x 37	256	8	253271	0.5464
9 x 38	288	9	226171	0.5461
26 x 15	320	10	195034	0.5587
57 x 7	352	11	196725	0.5168
28 x 17	384	12	162218	0.5634
55 x 9	416	13	154225	0.5585
18 x 31	448	14	139499	0.5687
54 x 11	480	15	131504	0.5687
68 x 9	512	16	127060	0.5602
18 x 38	544	17	115914	0.5710
23 x 32	576	18	107242	0.5805
48 x 16	608	19	102615	0.5764
25 x 33	640	20	96547	0.5826
19 x 45	672	21	95040	0.5716

Table 5 – Domain decomposition used to analyse the scalability for the PELAGOS025 model.

Each experiment, with a different decomposition, has been repeated 3 times with a total of 51 run. For each run we simulated 1 day with a time discretization of 1080 seconds per time step (a total of 80 time steps for each run). The table 6 reports the wallclock time

Cores	Wallclock (secs)
160	6220.90
192	5214.50
244	4589.58
256	3773.03
288	3275.46
320	2949.56
352	3100.42
384	2437.08
416	2445.70
448	2068.01
480	2109.46
512	2051.22
544	1688.38
576	1574.47
608	1620.28

640	1432.51
672	1379.29

Table 6 –Wall clock time for all of the configurations.

The speedup and the wallclock time functions are reported in the figure 14.

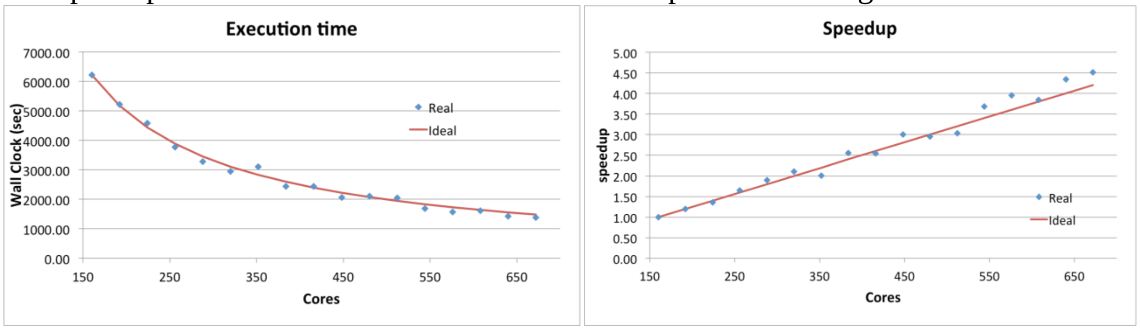


Fig. 14 – Wall clock time and speedup function.

The results report a super-linearity for several configurations. This behaviour is due to the different distribution of ocean points among processes when the level of parallelism increases. The reference configuration (with 160 cores) has worst balancing with respect to the others configuration. The figure 15 shows how the number of the ocean points for the heaviest process changes for different configurations. The red line reports the ideal distribution referred to the ocean points of the 160 cores configuration. Since the computing time is proportional to the number of ocean points, if, for a given configuration, the real ocean points are less than the ideal distribution, also the computing time will be less than the ideal and this produces a super-linear point in the speedup graph.

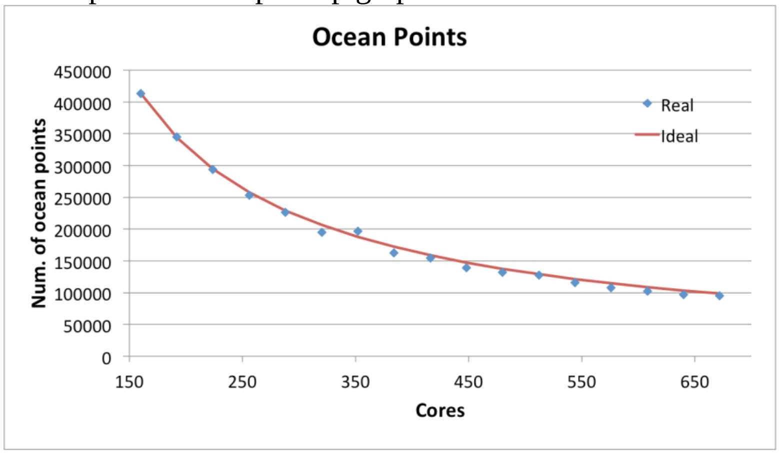


Fig. 15 – Number of ocean points for the heaviest process.

Finally, we have analyzed the code profiling and the parallel scalability for the most computing intensive routines. In table 7 we have: the elapsed time for the most computing intensive routines; the ratio with respect to the execution time of the whole model; the speedup and the efficiency.

	160		256		384		544		672	
	Elap. Time(s)	Elap. Time(%)	Elap. Time(s)	Elap. Time(%)	Elap. Time(s)	Elap. Time(%)	Elap. Time (s)	Elap. Time(%)	Elap. Time(s)	Elap. Time(%)
trc_stp	2856.15	45.34	1681.66	44.38	1032.85	42.43	685.41	39.51	531.81	38.17
tra_adv_muscl	2244.73	36.10	1355.00	35.76	849.51	34.90	585.52	33.75	454.48	32.62
tra_ldf_iso	446.82	7.19	260.23	6.87	165.28	6.79	113.09	6.52	92.01	6.60
tra_sbc	234.27	3.17	144.74	3.82	108.21	4.45	73.30	4.23	55.73	4.00
zps_hde	233.50	3.56	97.45	2.57	49.95	2.05	38.33	2.21	37.98	2.73
tra_zdf_imp	132.99	2.14	76.91	2.03	49.15	2.02	33.35	1.92	27.02	1.94
tra_adv	79.82	1.28	40.35	1.06	18.89	0.78	21.36	1.23	15.36	1.10

Table 7 – Routines profiling.

The most computing intensive routines are the *trc_stp* that represents the entry routine for the BFM model and the *tra_adv_muscl* that implements the MUSCL schema for the tracer's advection. Figure 16 shows the wall-clock time and speedup for this 7 routines. The speedup is super-linear due to the previously described motivation. In Figure 7, the computational weight of these 7 routines is reported and how it changes when the level of parallelism increases.

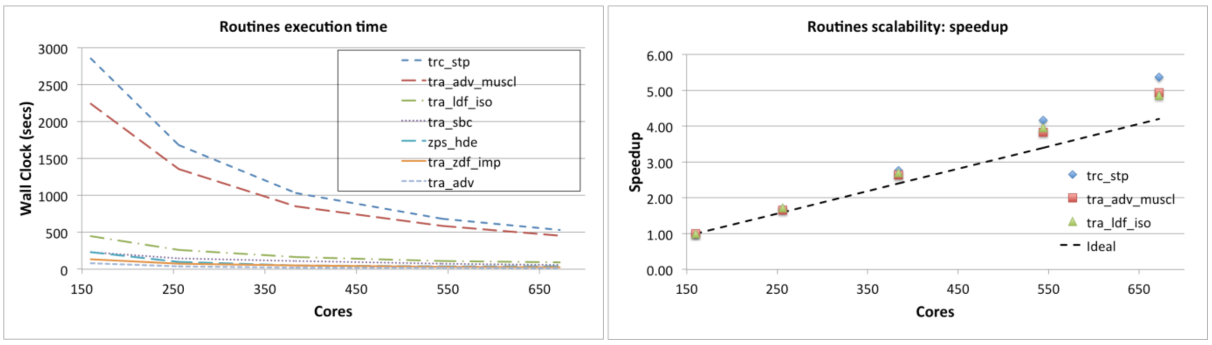


Fig. 16 – Routine's scalability: wall clock time and speedup.

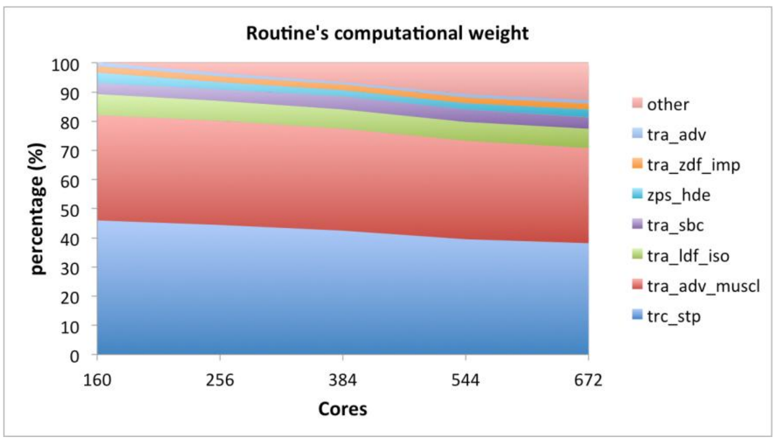


Fig. 17 – Computational weight of the most computing intensive routines.

Bibliography

[1] P. Michielse, J. Hill, G. Houzeaux, O. Lehto, and W. Lioen. Report on available performance analysis and benchmark tools, rep- resentative benchmark. Technical Report PRACE Project Deliverable D6.3.1.
[2] M. Tonani, N. Pinardi, S. Dobricic, I. Pujol, and C. Fratianni. A high-resolution free surface model of the mediterranean sea. Ocean Science, Ocean Sci., 4:1–14, 2008.
[3] P.Oddo and N.Pinardi. Lateral open boundary conditions for nested limited area models: a process selective approach. Ocean Modelling, 2007.
[4] Barcelona Supercomputing Centre. Paraver overview. BSC Performance Tools, 2010.
[5] D. M. Young. Iterative methods for solving partial difference equations of elliptic type. Trans. Amer. Math. Soc., 76:92–111, 1954.
[6] Barcelona Supercomputing Centre. Dimemas overview. BSC Performance Tools, 2010.

Porting and performance analysis of Cosmos-Millennium on Cray-XT5

Author: J. Silen, FMI

Introduction

Cosmos-Millennium

The model used is an application of the ECHAM5/MPIOM, version 1.2.0.1. The model is run in the complete "asob" mode on the institutes CrayFMI, Cray-XT5. It has also been run on CSC's similar machines and was successfully ported to SGI-Altix-4700 systems at LRZ, the Leibniz-Rechenzentrum in Bavaria, Germany, but no production runs were performed on either of them, due to a high load on their batch systems. The model consists of
1. Coupler: The coupler software is OASIS3, which manages the distribution of data between two or more interacting submodels. Here we used an atmosphere model, ECHAM5, together with a land and vegetation module, JSBACH and a model for aerosols, HAM, which were coupled to an ocean model, MPI-OM, supporting biogeochemistry through HAMOCC.
2. Atmosphere model: ECHAM5, supporting carbon cycle, JSBACH, and aerosols, HAM. Resolution 96x48, i.e. the T31L19 configuration.
3. Ocean model: The ocean is the MPI-OM, biogeochemistry HAMOCC. Resolution 120x96.
4. Atmospheric chemistry model: HAM, aerosols
5. Land and vegetation module: JSBACH, carbon cycle.

FMI Supercomputer Cray-xt5

Objectives

The main objective of this work is to identify and document the issues related with the portability and performance of the Cosmos-Millennium model on the FMI 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.

Description of execution and evaluation tests

Porting

The model is constructed to support the PRACE environment. Compilation and execution is controlled by ksh scripts which are assembled by other scripts using the m4 macro asssembler and a set of header files defining the compiler, the excecution and the machine environment. Once the definitions are done correctly the entire compile-execute cycle runs smoothly. Several compilers were tested on multiple platforms. On the Cray machines the PGI compiler was most convenient. On the SGI-Altix machines, the Intel compilers were used.

Execution

The compile cycle assembles scripts which set up a template for the execution process. The template is itself a script which is edited according to preferences and rerun to produce the actual jobs to be submitted to the batch processing system. Often the details of the operational batch systems differ and result in the need of directly editing also the submittable scripts accordingly. Both PBS and LSF batch systems were used. Detailes are provided only for the Cray-xt5 operated at the FMI.

Platform	
Execution platform	Cray-XT5
Environment	Module system with necessary modules available. **pgi** compiler **hdf5**, **netcdf** and **cdo** modules loaded.
Requirements	Module system with necessary modules available. **ksh** - korn shell (ksh93) **mpi1** - required for parallel execution **ftn** - a fortran compiler that supports an auto-double (or -r8) capability. OASIS3 requires Cray pointers. **netCDF** - the netCDF library for I/O **cdo** - for postprocessing integrated into the run-script.
Libraries	**Oasis**: netcdf **MPI-OM**: netcdf Standard Cray environment.
CPUs Used for each component	12 cores per node **ECHAM5: 5 nodes.** **OASIS:** 1 **node** **MPI-OM: 5 nodes**
Submit job command	"*qsub xxx.run*" to transfer the run script to the machine batch system In the script, "a*prun -n 1 oasis.x: -n ${nprocatm} echam5 : -n ${nprococe} mpi-om* " starts the parallell execution of the coupled code.

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

Optimization

Cray-xt5 has a convenient module system. If used properly standard compiler environments are set up automatically and reasonably well optimized code is produced. The main bottleneck is to ensure that the resources allocated to the atmosphere and the ocean models are well balanced and do not cause unnecessary waiting. The best performance was achieved by allocating slightly larger resources on the ocean model. The cray architecture requires an allocation of a complete node (12 cores) to OASIS3 although only one core is used.

Scalability

The best performance in wall clock time was achieved for 132 cores or 11 nodes. More cores slowed down the runtime performance. Without loss of performance, multiple jobs at the T31L19 resolution could be run in parallel. This was used in computing 30 x 45 years of forced simulations. All machine resources were used and the job was completed in about 4 days. The maximum throughput achieved was 43 simulated years/24 hours wall clock time. The allocation of resources to each submodel is important as it may influence the total time by 25%. Table 1. shows an example of allocating 7 nodes (84 cores, 72 cores for models and 12 cores for coupler) in different geometries to the models.

Other comments

During the porting and optimization work, it became obvious that the infrastructure available at different computing centers does differ significantly. In particular the job submission systems (PBS or LSF), use there own definition of resource allocation and does require manual work on the run scripts for the Cosmos-Millennium code. In the case of the SGI/Altix-4700 machine, the batch queue turn around complicated the optimization and porting work. To a lesser instant this was the case for the Cray-xt5 at CSC, where the development work could be made fluently on the front end machines. A challenge was to find out certain flags for allocating sufficient resources for the codes. The present work was very well supported by the installation of two Cray-xt5's at FMI. This made it possible for one month to use the systems during their testing period for this work and having several Cray Inc. enginers answering any questions. Thus two MPI related flags were included into the run script:

```
export MPICH_PTL_UNEX_EVENTS=200000
export MPICH_UNEX_BUFFER_SIZE=240000000
```

Without them, the jobs would start correctly but crash with a complaint of "unsufficient resources".

Summary

We were successful in porting and running the Cosmos-Millennium model using the coupled atmospheric and ocean models. The work was made on 3 platforms, two Cray-xt5's and one SGI/Altix-4700. Production runs were made on the former only. The performance shows an absolute wall clock minimum at this model resolution around 100 cores and this minimum is sensitive to the allocation of resources for computing the atmosphere and the ocean. In this setup one extended 1200 year experiment and one 30x45 year experiment has been performed.

Acknowledgments

I express my gratitude to the staff at CSC, LRZ and the Cray engineers supporting the

installation of equipment at the FMI. HPC work is magic based on years of experience.

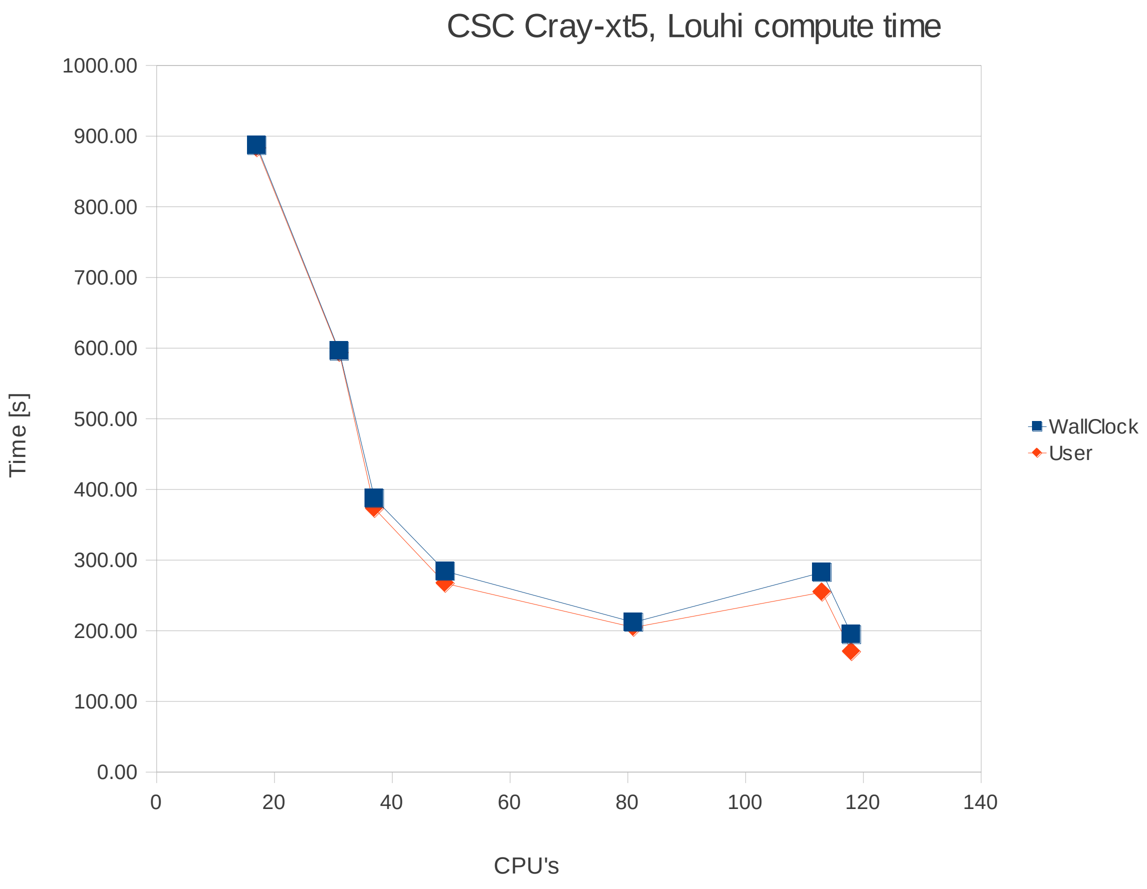


Figure 1: Demonstration of wall clock time as function of cores used on Cray-xt5 at CSC.

ECHAM5	MPIOM	TIME (s)
24	48	181
36	36	164
48	24	219
48	24	207

Table 1: Allocation of compute cores to ECHAM5 and MPIOM influences wall clock time given in [s] for computing a 1 month simulation. FMI Cray-xt5, December 2009.

Porting and performance analysis of Echam6 on Cray XT4

Contributors: Juha Lento, CSC, subcontractor to FMI (juha.lento@csc.fi)

Introduction

The study case is Echam6 atmosphere model and the CMIP5 - amip-LR model experiment at T63L47 resolution. The benchmark runs were done using CSC's Cray XT4 with AMD 2.3GHz Opteron processors. Current top of the line Intel Xeon processors deliver roughly 50 per cent application level speedup per processor core compared to the processors in Cray XT4.

Objectives

The objective is to study the parallel scalability of Echam6, test the I/0 server scheme introduced in Echam6 and to give feedback about the porting of the model.
Parallel scalability depends strongly on the size of the problem, i.e. the size of the computational grid. In this study we keep the size of the computational grid (T63 resolution) fixed, i.e. perform a strong scalability test.

Description of the execution and the evaluation of the tests

Porting and porting feedback

The model was compiled using PGI 11.2 compiler suite, with PGI "-fastsse" optimization level (except for sym1.f90), and with MPI and OpenMP parallelization. The stepon.f90 subroutine was modified to output timing information for each iteration step. The compilation was done by providing system dependent environment variables at the configure step, and then with a simple ‘make’.
The compilation of Echam6 is relatively easy, compared to the support libraries HDF5 and NetCDF, for example. The change that we would propose is to drop the Echam's internal C-language support library and use Fortran equivalent routines instead, or if not practical, use ISO-C binding available in modern Fortran versions instead of the cfortran.h interface.

Execution

The set up of the input files turned out to be next to impossible. In principle they are available, but to be really sure how to set the model parameters to match certain experiment run is not clear. There are shell scripts to achieve that goal, but the only thing sure about the shell scripts is that they will not work in a different environment from where they were written. The simple solution was to copy the contents of the work directory just prior to the execution of the model from the developer's environment to the current test environment. The set up of the experiment runs would not have been possible without the expert help from MPI-M Hamburg.
The benchmark run is one month simulation starting from the identical initial conditions at the start of the CMIP5 AMIP-LR experiment. The test run was repeated with varying the number of MPI tasks, OpenMP threads and the number of cores dedicated as I/O servers.

Results

The wall clock execution times of the parallel runs as a function of the number of the used processor cores for different MPI and OpenMP configurations are presented in table 1. The corresponding parallel efficiencies are presented in Table 2. The overall result is that the efficiency starts to drop significantly around 100 processor cores, and that OpenMP hybrid parallelization has relatively small effect on the scalability for the studied T63 resolution experiment. The results would likely be very different for larger resolution experiments with

larger computational grids.
The effect of the separate I/O server tasks (0, 1, 4 and 8 tasks) was studied, but the effect on the run times was found to be negligible for this particular experiment. The result is expected, as, in the studied experiment, the disk I/O is a minor contribution to run time.

	Processor cores						
Nthreads	4	8	16	32	64	128	256
1	18173	9156	4737	2453	1304	738	500
2	-	9947	5065	2643	1388	776	474
4	-	-	5690	2911	1550	848	514

Table1. Execution times from "time aprun echam6" in seconds as a function of the number of the used processor cores for different MPI task and OpenMP thread configurations.

	Processor cores						
Nthreads	4	8	16	32	64	128	256
1	100	99	96	93	87	77	57
2	-	91	90	86	82	73	60
4	-	-	80	78	73	67	55

Table 2. Parallel efficiency (percentage), relative to 4 MPI tasks with single OpenMP thread case.

Reducing the Sensitivity of the Met Office Unified Model to Rounding Errors

Contributors: O.R. Darbyshire, A.M. Clayton, Met Office.

Introduction

Numerical Weather Prediction (NWP) and Climate Prediction models are computationally intensive tasks involving a great many floating-point operations. As such, the results of these models are influenced by the nature of floating point arithmetic in a variety of ways. The binary system used on the majority of computer architectures is unable to represent floating point numbers with complete accuracy and there is some error introduced due to rounding. For a 64-bit floating point number on an IEEE 754 compliant system the upper bound of this error, often termed machine epsilon, is 1.11x 10^{-16}. Furthermore, the semantics of finite precision arithmetic are dissimilar to their exact counterparts and can introduce significant error (Knuth, 1981). The situation is further compounded as different architectures and compilers may introduce different rounding errors in intrinsic functions and their internal representation of floating point numbers. Even the level of compiler optimization can alter the results of floating point arithmetic.

The impact of these errors on the model results can manifest itself in several ways. Goel and Dash, 2007 studied the results of a spectral General Circulation Model (GCM) obtained on three different computer architectures and found the difference between the temperature fields on two architectures could be as large as 4.5°C. Modifying the format of the initial data given to the model from a free format read to one which gave the same level of accuracy on all three platforms greatly reduced the differences observed between the architectures. Rosinski and Williamson, 1997 studied the growth of temperature differences in the NCAR Community Climate Model (CCM) version 2 and found the growth to be faster than that which could be attributed to turbulent flow. The error growth was attributed to the accumulation of rounding errors and the response of discontinuous algorithms in the physical parameterizations to the evolving state of the model.

Rosinski and Williamson, 1997 also set out a framework that can be used to validate the port of a model to a new computer architecture. They describe three conditions that must be met:

- “During the first few times steps differences between the original and ported code solutions should be within one or two orders of magnitude of machine rounding.”
- “During the first few days the growth of a difference between the original and ported code solutions shout not exceed the growth of an initial perturbation introduced into the lowest-order bits of the original code solution.”
- “The statistics of a long simulation must be representative of the climate of the model as produced by the original code.”

In this work we describe the application of these porting tests to the UK Met Office Unified Model (MetUM). Firstly we document the growth of differences in the model due to an initial perturbation in the temperature field at the least significant bit. A methodology for reducing the sensitivity of the algorithms used in the physical parameterizations to perturbations in the initial temperature field is then outlined. Finally, results of the improvements made to the algorithmic design of the MetUM are discussed and the porting and validation of the model on a new architecture is described.

Model Overview

The MetUM is a highly flexible suite of numerical software for modelling the atmosphere that can be used across a wide range of length- and time-scales (Cullen, 1993). The model may be run in high resolution global and local area configurations for NWP and coupled to ocean and sea-ice models as well as earth system components for seasonal or decadal forecasting and climate prediction.

Unfortunately due to some of the algorithms used in the physical parameterizations employed in the model the MetUM code has been found to be particularly sensitive to small perturbations in initial conditions. These problems generally result from poorly conceived logic tests on real variables used to determine specific states in the model.

To take a simple example where there is the following test:

```
IF (SNOW_TILE(L,N).GT.0.0) THEN
TSTAR_TILE(L,N) = MIN( T_SOIL(L,1), TM )
END IF
```

where TM is 0°C. This ensures that the surface temperature TSTAR_TILE is at or below the freezing point if there is snow on the tile.

In a pair of runs differing initially by least-significant-bit perturbations, the following values were found:

Run 1: SNOW_TILE = 0.0,   T_SOIL ~ 5°C => TSTAR_TILE = ~5°C

Run 2: SNOW_TILE ~ 1.0e-19, T_SOIL ~ 5°C => TSTAR_TILE = 0°C

In both runs, the ground temperature is well above zero, and the intention of the coder is clearly that there should be no snow. However, in the second run an earlier calculation which should have removed all the snow from the tile has been affected by rounding error, leaving a snowflake or two behind. The consequence is a spurious jump of 5°C between the two runs.

It is errors such as these that result in a very large value of the RMS temperature difference between the control and perturbed run, as can be seen in Fig. 1. This makes the porting tests described previously difficult to carry out as the spurious and rapid growth of temperature differences may mask an underlying error in the porting. In an effort to reduce the spurious growth of errors software has been developed to track down these branching events at if tests so that the growth can be reduced and porting validations carried out with more confidence. This software is described in the subsequent section.

Methodology

The MetUM code is automatically instrumented in order to record the number of times each branch of an if test is executed. Once this has been done a control run can then be carried out and then compared to a run where the initial conditions have been perturbed at the least significant bit using random numbers sampled from a Gaussian distribution. Comparing the time evolution of the RMS Temperature difference between the control and perturbed runs gives a good indication of whether differential branching has occurred and the solutions are diverging.

The output from the software used to instrument the code can then be used to locate the routine where the branching occurred and the problem investigated further. A small N48 resolution global test job running on four processor cores is used when searching for spurious code branching.

Results

Reduction of Sensitivity to Initial Conditions

As mentioned earlier the over-sensitivity of the MetUM to perturbations in the initial conditions is a barrier to successful validation of the model when ported to new architectures. The branch tracking software developed has been used to track down coding errors which lead to differential branching in the code and hence large jumps in the temperature difference between the control and the perturbed runs. Figures 1 to 3 show the results of this work on three different systems: a Cray XE6 (HECToR), and IBM Power 6 and an Intel Xeon Linux machine. These figures show an ensemble of 20 runs with the thick red line being the average. In all cases the over-sensitivity of the MetUM trunk can be seen, with a large jump in the RMS of the temperature difference between the control and the perturbed run shown clearly in the first few timesteps. A modified version of the code, containing numerous fixes to reduce the sensitivity of the code to initial condition perturbations (r7271 on HECToR and r35820 on both IBM and Linux) exhibits much improved performance over the first 3 hours of a model run.

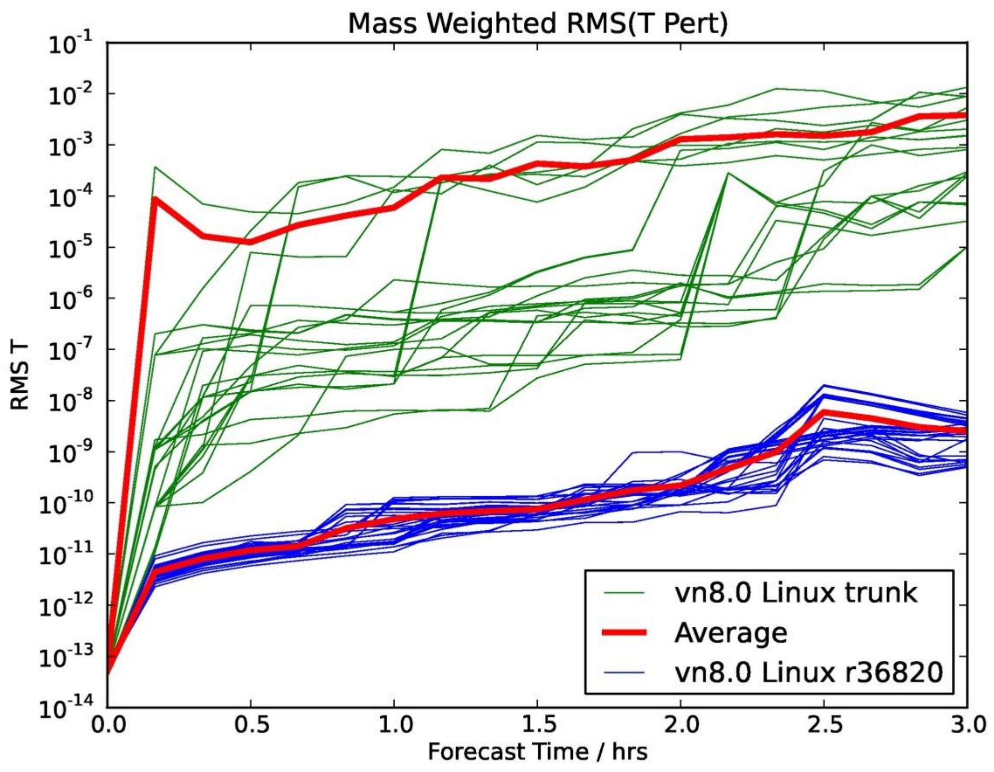


Figure 7 RMS of the temperature difference between the control and perturbed run of an N48 model on Linux. Results of an ensemble of 20 runs are shown using the trunk of the MetUM code and a version containing fixes for spurious branching, r36820.

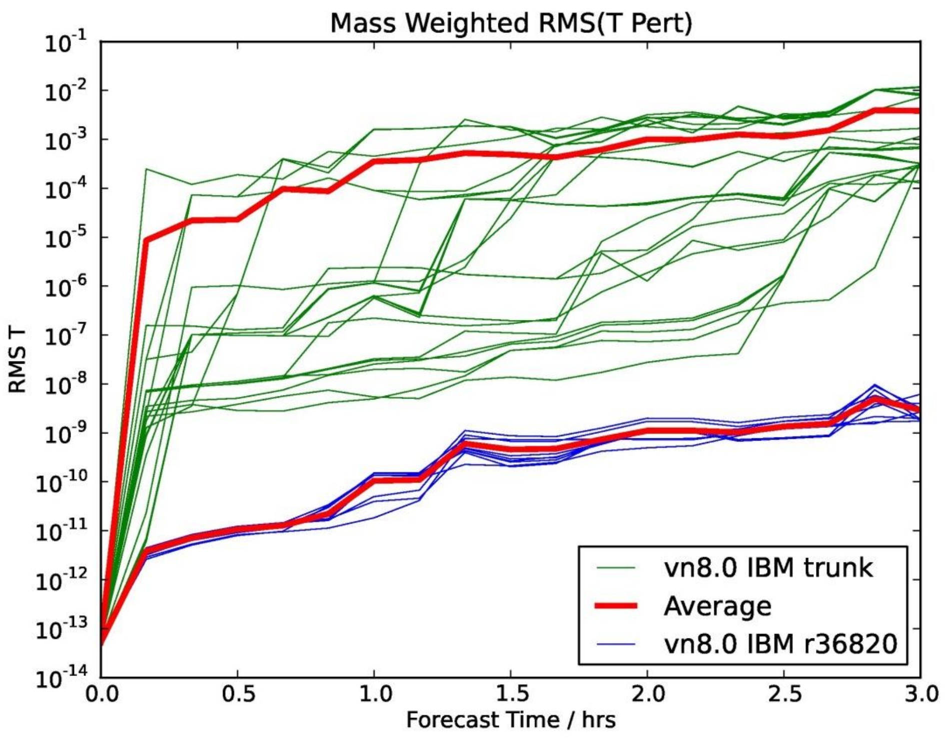


Figure 8 RMS of the temperature difference between the control and perturbed run of an N48 model on the IBM Power6. Results of an ensemble of 20 runs are shown using the trunk of the MetUM code and a version containing fixes for spurious branching, r36820.

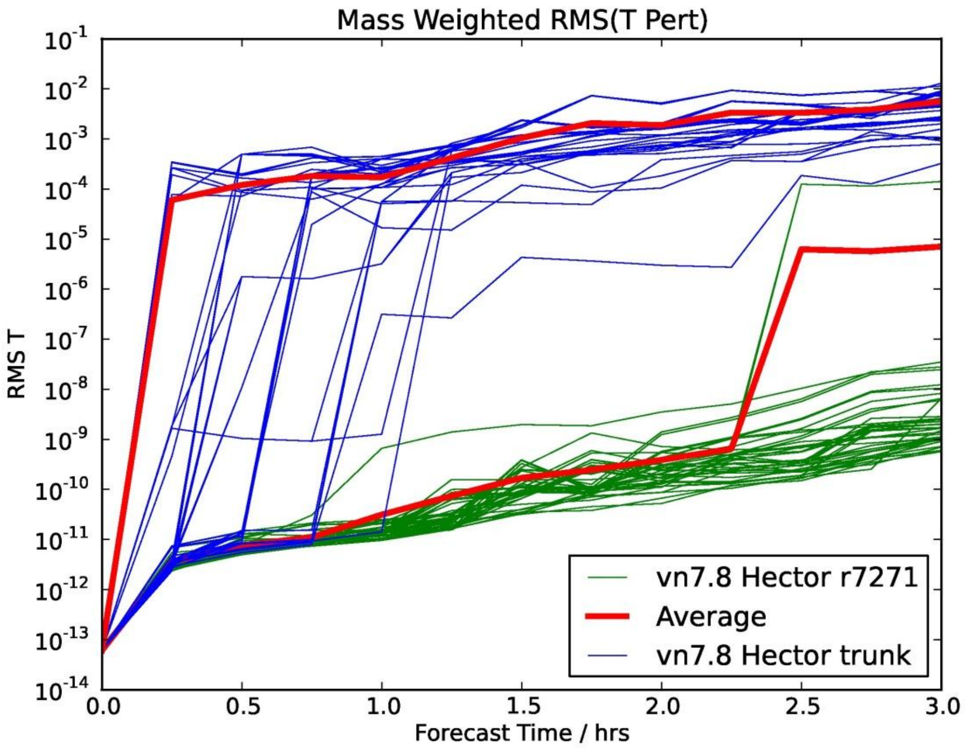


Figure 9 RMS of the temperature difference between the control and perturbed run of an N48

model on HECToR. Results of an ensemble of 20 runs are shown using the trunk of the MetUM code and a version containing fixes for spurious branching, r72 71

This improvement in the model's response to initial condition perturbations will allow the porting tests described in the introduction to be carried out with significantly more confidence than previously possible. Similar tests can also be carried out to assess the impact of compiler optimizations and new algorithms on the performance of the model.

Model Port Validation

The performance of the model is now such that it is possible to complete the first porting test when running the MetUM code on HECToR. Figure 4 shows RMS of the temperature difference for the same ensemble of runs on both a Linux machine and HECToR. The blue lines are the RMS temperature difference between the Linux runs and Linux control and the green are the RMS temperature difference between the HECToR runs and Linux control. Here the lower growth of perturbations detailed in the previous section can still be seen and the differences between the runs on the two machines are minimal. The growth of perturbations is close to the level of machine rounding in most of the cases, however these is still some room for improvement in this area. Due to a lack of computational resources on HECToR the other tests were not attempted.

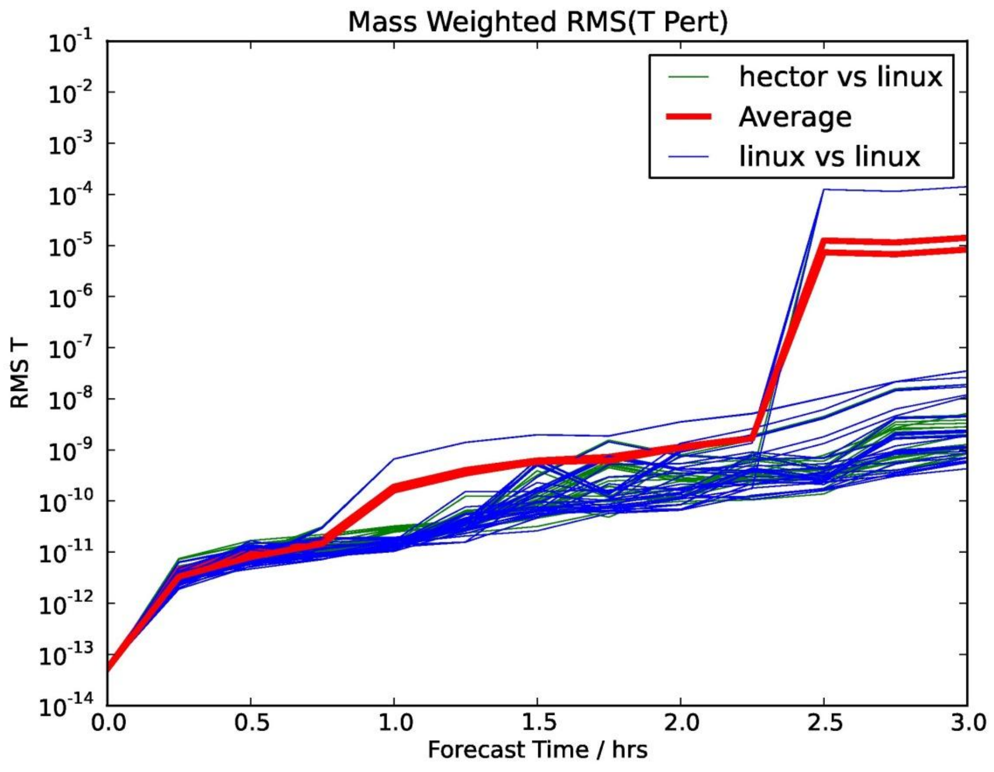


Figure 10 RMS of the temperature difference between the Linux control and Linux perturbed run of an N48 model (blue) and Linux control and HECToR perturbed run (green). Results are shown using r7271 of the MetUM code.

Optimization Error

The same strategy used to expose errors porting the code to a new system can be used to test potential new additions or optimizations to the code. Figure 5 shows two ensembles of runs on an IBM Power6 using the N48 model employed previously. The run in blue is compiled with the option -qstrict=exceptions which disables all transformations likely to affect exceptions or be affected by them (IBM, 2011). but enables other optimizations that might affect floating point semantics. The run in green is compiled with -qstrict which disables all semantics-changing transformations (IBM, 2011). The difference between the two runs is obvious and this method represents the beginnings of a system for testing and catching the introduction of such errors into the code.

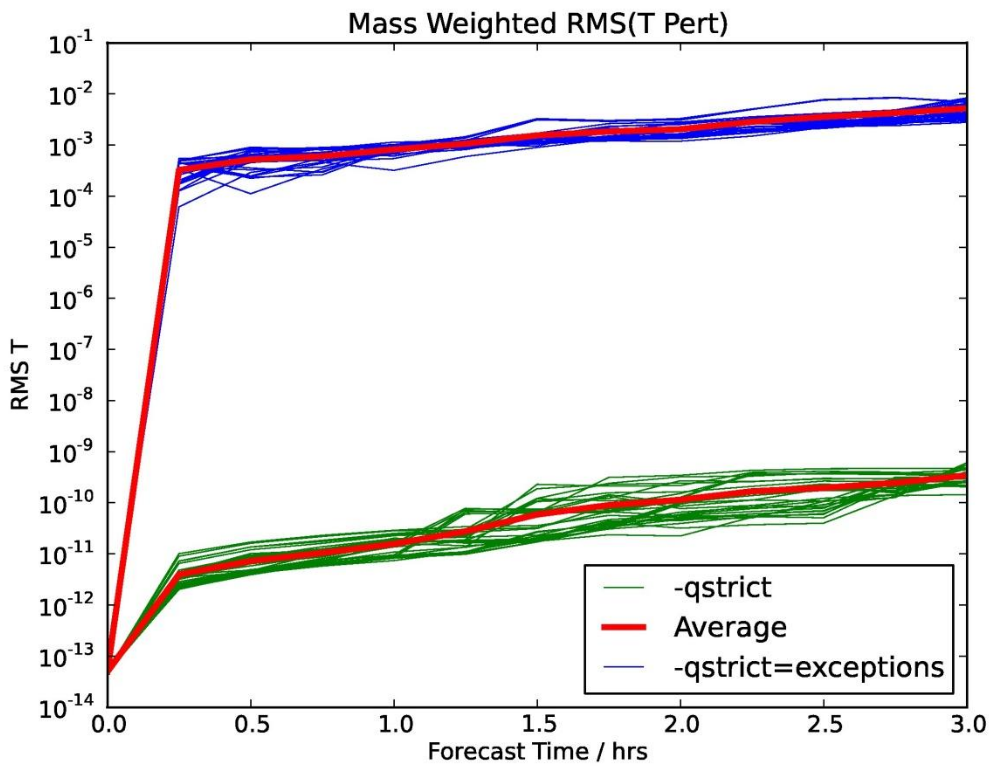


Figure 11 RMS of the temperature difference between the control and perturbed run of an N48 model on the IBM Power 6. Results of an ensemble of 20 runs are shown using an executable compiled with -qstrict=exceptions (blue) and -qstrict (green).

Summary and Conclusions

A system for tracking and diagnosing spurious code branching, a source of model divergence, in the MetUM has been developed. The rate of growth of a small perturbation has been significantly reduced, allowing some of the porting tests suggested by Rosinski and Williamson, 1997 to be completed. The improvements should increase the ease of porting the model to new architectures in the future and allow other coding or optimization changes to be tested to see if they alter the sensitivity of the model.

References

M. J. P. Cullen. The unified forecast/climate model. Meteorological Magazine, 122(1449):81–94, 1993.
S. Goel and S. K. Dash. Response of model simulated weather parameters to round-off-errors on different systems. Environmental Modelling & Software, 22(8):1164–1174, 2007.
IBM. XL Fortran for AIX v12.1. IBM, 2011.
D. E. Knuth. The Art of Computer Programming (Volume 2). Addison–Wesley, 1981.
J. M. Rosinski and D. L. Williamson. The accumulation of rounding errors and port validation for global atmospheric models. SIAM Journal on Scientific Computing, 18:552, 1997.

UM Scaling on HECToR

Contibutors: O.R. Darbyshire, Met Office.

Introduction

In more recent releases of the UM the amount of OpenMP parallelisation has been increased in order to take advantage of the decrease in execution time that can be achieved using a hybrid MPI-SMP approach. This approach is especially beneficial for codes whose scalability is limited by memory bandwidth such as the UM.

Methodology and Results

Four configurations of a N512 HadGEM3-A (atmosphere only) model using UM7.8 have been tested on HECToR in order to find the most efficient configuration in terms of atmosphere decomposition and number of OpenMP threads. Three of these models were analysed with Scalasca to obtain a more detailed picture. Both instrumented and un-instrumented executables were run for 24 hrs of model time.
Table 1 details the decomposition of the atmosphere and the number of OpenMP threads on 6120 processors (i.e. using 6120 MPI processes, one per processor). The following three columns give the maximum time in seconds a single process spent executing model code, was idle waiting for OpenMP work and the output from the UM timer routine. This information is obtained from the Scalasca instrumented runs. The final column gives the UM timer output, again a maximum of all the processes, for an un-instrumented run. Values labelled approx are an estimate as the runs hit their wall clock limit.
There are two main points to note: the model is fastest when using two OpenMP threads and there is a large overhead incurred to use Scalasca, approximately 300% of the un-instrumented time.
Table 2 shows the total time spent by every process executing different categories of code: time spent in instrumented MPI calls, time in OpenMP API calls and code generated by the OpenMP compiler, idle time spent by OpenMP threads. The table also shows the total amount of MPI data transfer. To be clear, the table shows the sum of the time spent in each process for each category. The data in table 1 and table 2 are related. The sum of the times in each row of table 2 divided by the number of MPI processes used (6120) gives the execution time listed in table 1. In table 2, for all the runs the total amount of time spent executing model code remains quite similar (within approximately +/-5% of the 3 OMP thread figure). However, as the number of OpenMP threads is increased the time spent in MPI calls is reduced but the amount of OpenMP idle time increases. In fact the rate that the OpenMP idle time increases as the number of threads goes up is about 4 times the rate at which time spent in MPI calls reduces.

λ pts	φ pts	OMP thds	Execution (s)	Idle OMP Threads (s)	UM Timer Wallclock (s)	UM Timer without Scalasca (s)
102	60	1				2000 (approx)
60	51	2	3907	2718	3851	945
40	51	3	4428	3209	4406	1071
30	51	4	4927	3638	4898	1400 (approx)

Table 1: Maximum of all processes from both instrumented and un-instrumented runs.

λ pts	φ pts	OMP thds	Execution (s)	MPI (s)	OMP (s)	Idle OMP (s)	MPI data transfer (TB)
60	51	2	9.9230×10^6	5.4109×10^6	4.2404×10^5	8.1532×10^6	75.92

40	51	3	9.4553×10^6	4.0560×10^6	8.2290×10^5	1.2767×10^7	61.58
30	51	4	9.3650×10^6	3.2668×10^6	1.2376×10^5	1.6290×10^7	54.67

Table 2: Total times for all processes for the Scalasca instrumented runs.

These runs should be considered an approximation of the true performance for two reasons: first, although the runs must read the start dump they do not write any dumps to disk. Second, at the beginning of the run the solver hits the limit of 200 iterations for about 30 time steps before settling down, see Figure 1. If this peak only occurs at the start the true time required per day of a longer run will be significantly less than suggested by these runs.

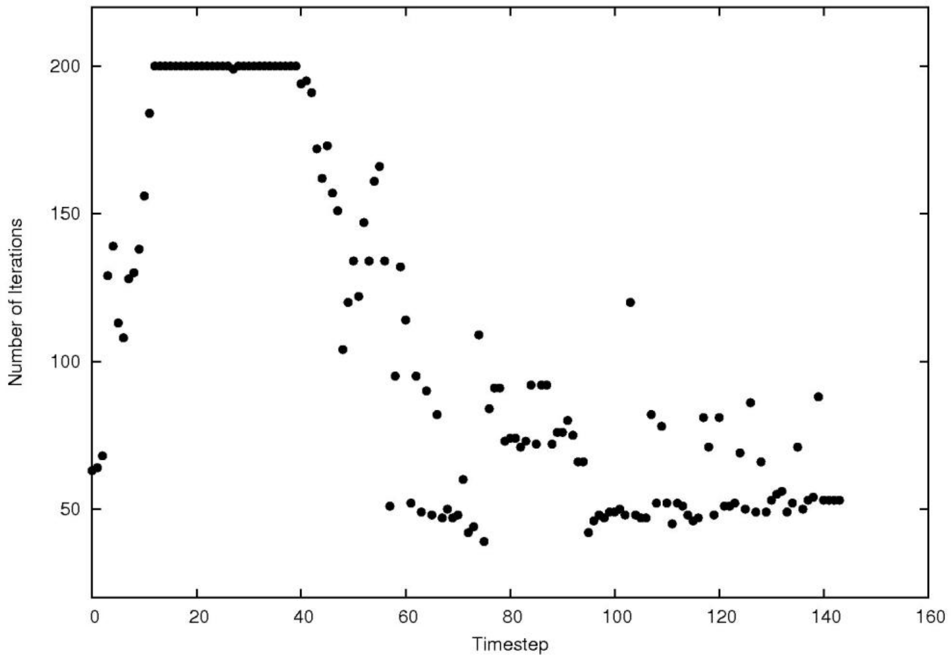


Figure 12 Iterations required to reach the convergence criterion at each time step. Hard limit at 200.

UM 8.0 setup for HECToR and HERMIT

Getting UM version 8.0 setup on HECToR was driven by the PRACE work. Originally vn8.0 was setup using the Pathscale compiler which was the default choice for running the UM on HECToR at the time. The amount of OpenMP code proved a problem as the compiler was buggy with nested OpenMP calls and work was required to trace the routines causing problems and reduce the optimisation level or turn off the OpenMP. The upgrade of HECToR to the new AMD interlagos chips coincided with the retirement of the Pathscale compiler and the default compiler for the UM on HECToR became the Cray compiler.

Working alongside the NCAS-CMS team in Reading work was carried out to move the UM on Hector to the Cray compiler (cce). This involved a substantial level of debugging are numerous routines required different levels of optimisation in order to run. This was further compounded by the need for the N512 model to run with more than one OpenMP thread, something not supported initially by NCAS-CMS.

Small changes were also required for GCOM in order to build with the cce. The call to MPI_init had to be changed to MPI_init_thread and another routine to get the maximum value an MPI tag can tag had to be altered to call a deprecated version as the new one would not work properly on the phase 3 version of HECToR or on HERMIT.

Version 8.0 of the UM is now setup and running on both HECToR and HERMIT. Performance of the model on HERMIT is better than expected and it scales well to 4000 cores. Further work is required to iron out some issues with the I/O server and STASH meaning. A contractor from Cray has been employed to assist and look into these issues. He will also look at:

- MPI tuning
- Scaling with different numbers of OpenMP threads.
- Debugging tools
- I/O server tuning

Summary and Conclusions

Work has been carried out on HECToR to investigate the OpenMP scalability and to port the model to the new Cray compiler. At UM 7.8 it was found that running with 2 OpenMP threads gave the best performance. The porting work at version 8.0 was successful although there is room for further performance improvement with careful tuning. The amount of OpenMP parallelised code in version 8.0 has increased significantly hence there may now be benefit from running with increased numbers of threads. Recent scaling work on HERMIT has shown four threads to give good performance.

Technical issues with the UPSCALE PRACE project

Contibutors: S. Mullerworth, Met Office.

This section is based on interviews with UPSCALE team members about some of the technical issues that arose during the UPSCALE project jointly run by the Met Office and NCAS. The motivation for writing this section is not to go into the detail of the technical issues, but to list the issues that are likely to arise and should therefore be considered when planning future similar projects. Detailed information can be found in "High resolution climate modelling; the UPSCALE project, a large simulation campaign" Mizielinski et. al, in preparation for submission to Geoscientific Model Development.

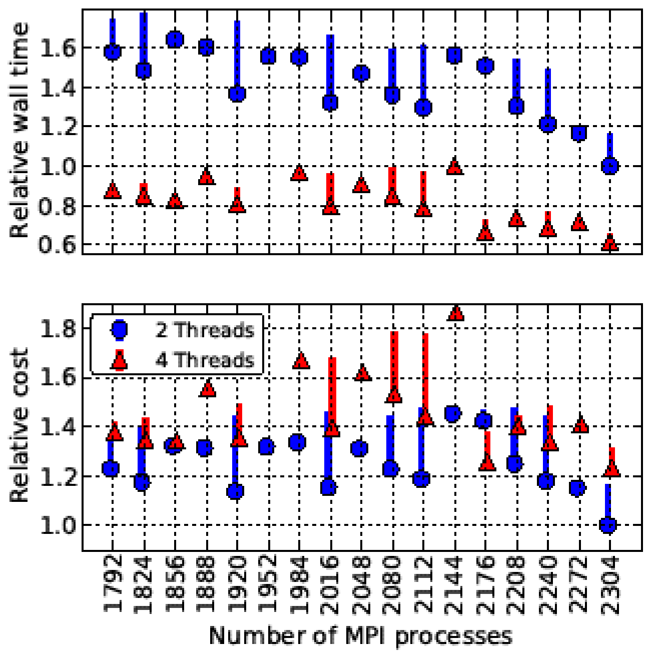
UPSCALE ran a series of high resolution (mainly 25km, but also some 12km resolution) configurations of the Met Office atmospheric model on the HERMIT supercomputer within a PRACE project. The project used 144 million core hours of HERMIT over one year and generated 300 Terabytes of data.

The core of the UPSCALE project was a series of 25 model-year runs each of which ran on up to 9408 Cray XE6 cores. At a 25 km resolution, the Met Office Unified Model simulated 6 model-months per day. At its peak, the UPSCALE project was running 5 simulations concurrently, so using 47 thousand cores of HERMIT.

Prior to the UPSCALE project, the Met Office and NCAS have had experience porting the model to other large supercomputers, such as the Earth Simulator.

All models and systems are different, so the experiences below are summarised to give an indication of technical issues that are likely to arise with other similar projects, and to help other groups to plan for issues that may arise.

- **Project resource request:** Climate models generally request access on Tier 0 PRACE machines because the resources available elsewhere limit the resolution at which the model can be run. Necessarily, this means that it is difficult to estimate the resources required to complete an experiment before access to the machine is enabled. Furthermore, the particular properties of the Tier 0 platform can work well for the model or can work poorly meaning that the resource request can be over or under-estimated.
- **Initial porting:** Input from Cray analysts was critical to the successful porting and tuning of the model. Advice from Cray analysts also helped to optimise the model significantly, speeding up the model by a third. Two key improvements were:
    - The IO was tuned by ensuring suitable placement of the IO servers within the system. For example, ensuring each IO server ran on a different node
    - The processor configuration was tuned by scanning around 100 different model decompositions. In addition to finding some processor counts that gave widely differing performance on different decompositions (the bars in the diagram below illustrate the range of time for differing decompositions with the same core count), this work also found that an example where, for example, the core count was increased by 7-8% from 2144 to 2304 to gave a 30% cost reduction:



- **Model-specific IO issue:** The Met Office Unified Model has a post-processing “climate means” subsystem that generates seasonal and annual means on the fly. Unfortunately, within the PRACE configuration, this system disproportionately impacted the amount of IO being generated. As a result, the system could not be used and the outputs had to be generated in another way. While the climate means system is specific to the Met Office Unified Model, it is given as an example of an unexpected delay in porting that can arise in any project.
- **Model stability issues 1:** It was found that the high resolution version of the model was, relative to the standard resolution version, more prone to instabilities such as grid-point storms, resulting in the jobs needing to be regularly repaired and restarted. Before during and after this project, work has continued to try and improve stability such that failure rates have dropped from an original rate of once per 9 months to once per 19 months, and the latest incarnation has not suffered any failures so far.
- **Model stability issues 2:** As a time-slice run, this model was run with climates similar to current climatologies, and with warmer climates. Instabilities were also higher in the warmer climates, probably because the model configuration had not been tested thoroughly against that climatology. Analysing these stability problems was made more difficult by the lack of appropriate tools. Existing tools to plot data did not have the ability to deal with the very fine grids.
- **Model versus Machine stability issues:** Often it was difficult to detect the difference between issues caused by hardware failures and model stability issues. Model stability issues could be identified by checking iteration counts in the solver or monitoring vertical velocities. Evidence of machine instabilities, however, relied on monitoring which nodes the models that failed were running on so as to identify faulty nodes.
- **IO performance:** IO was very intensive in the model simulations. On occasions, total IO load on the system caused model slowdowns resulting in jobs running out of time in the queue.

- **Job prioritisation:** There was no ability to prioritise one job above another within the project on the HERMIT machine. It would have been beneficial to the project to allow some of the runs to have a higher priority than others.
- **Data transfers:** Climate models generate large amounts of data; the amount of data was a big surprise to the systems teams. The UPSCALE project generated 300 Terabytes in total. The PRACE rules require that the data is removed from the host machine at the end of the project. Therefore a significant subproject of UPSCALE related to the transfer of the model data to BADC. Initially, gridftp was not available, and the multiple rsync processes that were being used caused issues with HERMIT. Furthermore, there were issues with transferring some of the files due to their large size: they had to be split into smaller files and rejoined at BADC. When gridftp became available, further delays were caused by the need to obtain eScience certificates. Eventually, however, transfers at the rate of 2-6 Terabytes per day with built-in checking were running reliably, and all data was removed from HERMIT in good time. That said, monitoring the transfers and resolving issues required human input a few times a week over a period of some months.
- **Data Storage 1:** Clearly, planning for the storage of 300 Terabytes is a significant undertaking. Aligning a bid for computing resources within one system that may or may not be successful with a bid to store a large amount of data in another system is a difficult issue. In this case, the UPSCALE team were extremely fortunate that the project coincided with the introduction of the new JASMIN storage system at BADC. Prior to JASMIN coming on line the data was being stored on a range of different servers which was making the data difficult to analyse, and not all the data was backed up. JASMIN provided space to securely archive all the data in one place. It has been said by members of the UPSCALE team that the UPSCALE project was as dependent on JASMIN as it was on HERMIT.
- **Additional considerations:** it should be noted again that UPSCALE involved running an atmosphere model only. Addition of other components such as an ocean model will add additional issues by doubling up the effort required to port and tune models, by adding an extra technical issue related to coupling the two models efficiently, by needing to support the deployment of the a coupled model efficiently on the architecture and by needing to load-balance the deployment.

PART III: Concluding remarks

Summary of the results

The deliverable reports results obtained from the following coupled models ARPEGE-NEMIX, IPSL-ESM and EC-Earth, METOFFICE Unified Model, Cosmos-Millennium) and one stand-alone models (NEMO) tested on a number of PRACE HPC infrastructures within the IS-ENES network. 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:

- IPSL-ESM coupled model:
    - Under the PRACE preparatory access grant, COUAC, the IPSL model has been ported to CURIE and comparisons with the deployment on JADE (CINES machine) made. A high resolution version of the model has been ported and is now entering a performance tuning phase.
    - To support the porting to the new machine, the IPSL compilation and execution environment has been ported to CURIE and the plan is to install the IPSL running environment shortly to support production runs.
    - There is still an issue running MPI parallel ocean models and hybrid MPI-OpenMP atmosphere models on the machines currently available. The Computing Centres must be encouraged to address this with urgency.
- ARPEGE-NEMIX coupled model validation on PRACE machines:
    - In further work with IS-ENES ESMs on PRACE machine, the validation version of the ARPEGE-NEMO coupled model has been ported to PRACE tier-0/tier-1 machines and a successful porting validation is reported with comparisons to previous implementations on NEC and SGI Altix platforms. This work prepares the way for a port of a full version of NEMO to the PRACE machines.
    - Under a PRACE preparatory access project, work has been undertaken to enhance the performance of the OASIS coupler on the PRACE machine CURIE, using the new OASIS-MCT implementation. Some science issues with the port were identified and corrected and work towards a high resolution port for production use is now in progress.
- EC-Earth coupled mode on CURIE
    - A high resolution version (T799 and ¼ degree Ocean) of the EC-Earth ESM was successfully ported to and benchmarked on up to 3500 MPI processes on CURIE (an Intel Xeon plus Infiniband cluster) by LIU.
    - At this scale, the efficiency was approximately 50% of the most efficient run. An analysis of the behaviour or timesteps with and without I/O was investigated revealing that, though the I/O time varied timestep to timestep, the overall scaling of I/O timesteps is consistent with timesteps in which only computation occurs.
    - Work to create a standard environment which can be installed on a new machine to ease the porting process is also reported.
- EC-Earth coupled model on MareNostrum

- The experience of porting EC-Earth and other models has identified some ‘best practice’ that could be followed. EC-Earth consists of a number of component models (IFS, NEMO), the OASIS coupler and a number of essentially standard libraries (MPI, OpenMP, NetCDF). Increasingly, the standard libraries are already available on a new machine (and they – including, possibly a number of commonly used versions - should be made available by the centre housing the machine). Porting the component models reveals that some (versions of) models are more easily ported than others. Porting NEMO and OASIS to MareNostrum was relatively simple because each has centralised, in the source file structure, issues that typically cause porting to be difficult. The version of IFS used (that used in EC-Earth V2) was difficult to port and porting issues were not located in a centralised place. EC-Earth, V3 is much improved in this respect. This centralisation is a mechanism that has been developed in a number of community models, such as WRF and CMAQ, and lessons learned from these community codes should be heeded in ESM model development.
- Several version of EC-Earth were successfully ported to MareNostrum, including a version with a recent update to NEMO (PELAGOS025), and examples of load imbalance and serialisation within IFS and NEMO identified and analysed using Paraver. Several options for performance improvement were identified and evaluated. However, scalability is strongly limited by the current numerical schemes implemented.

- NEMO stand-alone model:
  - CMCC has undertaken porting, benchmarking and optimization work with NEMO on Marenostrum. Profiling revealed several target routines for performance improvement typically to reduce communication costs (either changing communication patterns of overlapping communication with computation). Also, improvements to the handling of I/O have been made. Some work to demonstrate the possibility of a hybrid OpenMP-MPI solution also shows promise.
- FMI Cosmos-Millennium on Cray-XT5
  - A successful porting and running exercise with the coupled atmosphere and ocean models of the Cosmos-Millennium model has been completed.
  - Three platforms were involved: two Cray-XT5s and one SGI/Altix 4700 with production runs being undertaken on the Cray machines. The minimum wallclock performance for the model (resolution T31L19) occurs at around 130 cores and is sensitive to the allocation of resources to the atmosphere and ocean.
  - One extended 1200 year experiment has been run with this configuration and an ensemble of thirty 45 years runs executed (with each ensemble member running on the optimum number of cores for an instance of the model).
- CSC (subcontractor to FMI) ECHAM6 atmosphere on Cray-XT4
  - A strong scaling case study of the Echam6 atmosphere model and the CMIP5 - amip-LR model experiment at T63L47 resolution has been performed on a Cray-XT4. Hybrid MPI-OpenMP parallelization was used and run-time and efficiency results presented.

- Porting difficulties were identified with the support libraries (HDF5 and NetCDF). The source of these problems was identified as the use of ‘old’ C interface language techniques. It is suggested that moving to either Fortran-based interface routines, or using the modern Fortran standard interface mechanisms to C would alleviate these problems.
- The scripts used for input file set up were found to be essentially non-portable to the Cray environment, and a workaround - based on copying the initial state from a development directory - was implemented.

- METOFFICE Unified model (UM)
  - When porting the UM, it has been found that the code behaviour is very sensitive to the results of floating point operations, leading, for example, to spurious branching resulting from floating point comparisons. Work has been undertaken to restructure the code to reduce its sensitivity in this respect, thus improving the porting process (in terms of validation of the science computed).
  - A system for tracking and diagnosing spurious code branching, a source of model divergence, in the UM has been developed. The rate of growth of a small perturbation has been significantly reduced allowing some standard porting tests (suggested by Rosinski and Williamson in 1997) to be completed. These improvement should increase the ease of porting the model to new architectures in the future.
- METOFFICE Unified model:
  - Four configurations of an N512 HadGEM3-A model were tested on HECToR to investigate the most efficient configuration of MPI processes in an atmosphere decomposition and the number of OpenMP threads to use in a hybrid MPI-OpenMP implementation. This work was driven by interaction with PRACE.
  - Comparisons show that two threads provide the best performance improvement (the mapping of processes and threads to cores used in the tests is not explicitly stated).

The following table brings together scalability results for several of the models reported in this document.

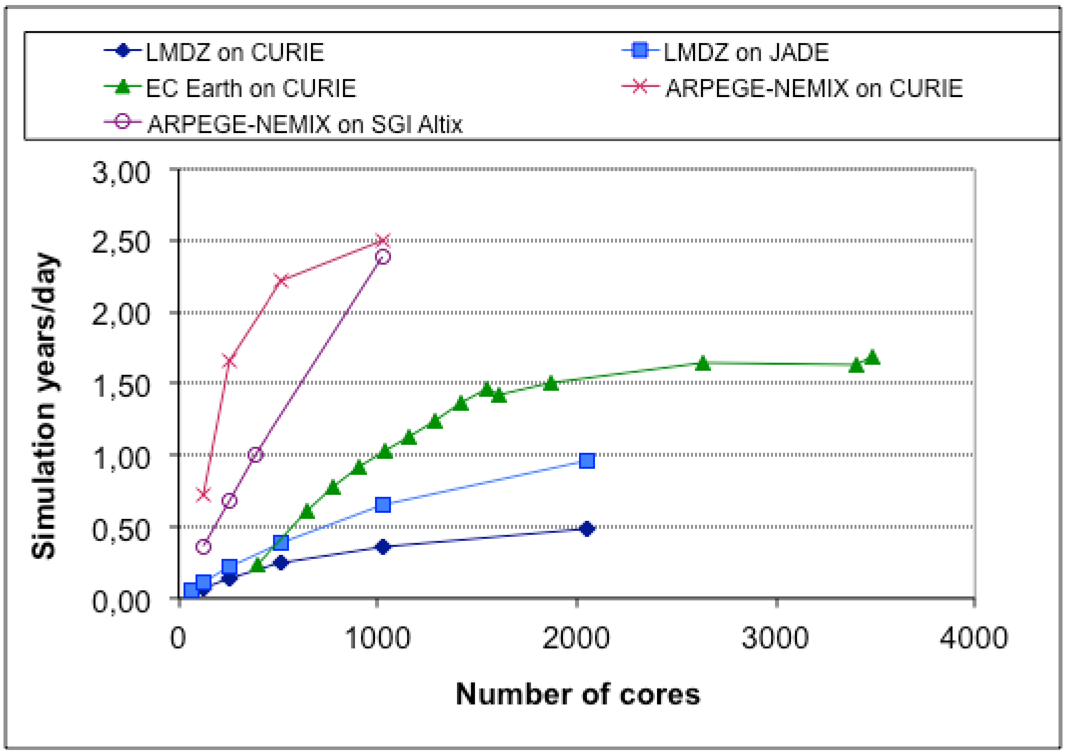


Table 1: Simulated years/day vs number of cores for several of the European ESMs reported in this document. Resolution between 25 and 30 km. (Note, the models do not necessarily compute comparable science etc.)

All results presented in the table have been translated into the (absolute) metric of 'simulated years per day'. Care should be taken when drawing conclusions from such a graph as the models reported differ in several crucial aspects, including science content, resolution, I/O use etc.

The graph shows that most European models, at 25-30 km resolution, currently scale to around 1000-3000 cores. This conclusion gives impetus to the current initiatives being undertaken to address scalability limitations in the major European ESMs.

One recent European initiative of note, is the recent work led by Pierre-Luije Vidale (University of Reading), using the HiGEM version of the Met Office's UM on the Cray-XT6 (HECTOR). This work shows HiGEM scaling to around 12000 cores. Work in the US also shows models scaling to tens of thousands of cores. This work, and other related initiatives in Europe and the US, are summarised in the presentation at SC12 given by Sylvie Jussaume, "Modelling the Earth's climate system: data and computing challenges", SC 2012, Salt Lake City.

Scalability well beyond tens of thousands of cores (and threads) will be required for the Exascale computers envisaged to be in production towards the end of the current decade.

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].