Performance Analysis of EC-EARTH 3.1

John Donners[a], Chandan Basu[b], Alastair McKinstry[c], Muhammad Asif[d], Andrew Porter[e], Eric Maisonnave[f], Sophie Valcke[f], Uwe Fladrich[g]

[a]*SARA, Amsterdam, The Netherlands*
[b]*NSC, Linköping, Sweden*
[a]*ICHEC, Galway, Ireland*
[d]*IC³, Barcelona, Spain*
[e]*STFC, Daresbury, United Kingdom*
[f]*CERFACS, Toulouse, France*
[g]*SMHI, Norrköping, Sweden*
Please send any comments or questions by email to john.donners@sara.nl

Abstract

The EC-EARTH model is a global, coupled climate model that consists of the separate components IFS for the atmosphere and NEMO for the ocean that are coupled using the OASIS coupler. EC-EARTH was ported and run on the Curie system. Different configurations, using resolutions from T159 (approx. 128km) to T799 (approx 25km), were available for benchmarking. Scalasca was used to analyze the performance of the model in detail. Although it was expected that either the I/O or the coupling would be a bottleneck for scaling of the highest resolution model, that is clearly not, yet, the case. The IFS model uses two MPI_Alltoallv calls per timestep that dominate the loss of scaling at 1024 cores. Using the OpenMP functionality in IFS could potentially increase scalability considerably, but this does not yet work on Curie. Work is ongoing to make MPI_Alltoallv more efficient on Curie. It is expected that I/O and/or coupling does become a bottleneck when IFS can be scaled further than 2000 cores. Therefore, the OASIS team increased the scalability of OASIS dramatically with the implementation of a radically different approach, showing less than 1% overhead at 2000 cores. The scalability of NEMO was improved during an earlier PRACE project. The I/O subsystem in IFS is described and is probably not easily accelerated unless it is being rewritten and uses a different file format.

1. Introduction

EC-EARTH is a project, a consortium and a model system [1]. The EC-EARTH model is a state-of-the-art numerical earth system model (ESM) based on ECMWF's Seasonal Forecasting System. Currently, the EC-EARTH consortium consists of 24 academic institutions and meteorological services from 11 countries in Europe. EC-EARTH Version 3, which is the most recent version, is a coupled ESM comprising ECMWF's atmospheric model IFS, including H-TESSEL land model, the general ocean circulation model Nemo, the LIM sea ice model, and the OASIS coupler. The developers of EC-EARTH are keen to collaborate on scaling up their application. Several groups that are willing to work in PRACE Work Package 7 Task 7.2e “Weather, Climatology and Earth Sciences”

already contribute to EC-EARTH (SNIC-KTH) or either IFS (ICHEC) or NEMO (SARA). EC-EARTH is used in many EU FP7 projects, including THOR, COMBINE, EUCLIPSE, ECLISE and IS-ENES. The model is used for developing research on understanding the earth system and for developing climate scenarios. The model output will contribute to IPCCs 5th Assessment. ECMWFs IFS atmosphere model and the NEMO ocean model are the main components of EC-Earth. A variety of resolutions is used. High resolution runs targets the simulation of physically realistic climate phenomena. Coarse resolution runs are done in ensembles (ie many, slightly different simulations) in order to capture statistical uncertainty. Ideally, high resolution simulations need to be performed in an ensemble sense. A discussion of reliability of the overall computation and its sensitivity to input data error can be found in the 'Papers' section on the website http://ecearth.knmi.nl. A discussion of the limitations of the ambitious goal on long-term predictions can be found in the general, scientific literature about climate simulation.

1.1. IFS

ECMWF is the European Center for Medium-range Weather Forecasts and runs the global weather prediction model IFS every 12 hours at very high resolution (at the moment 16km). The IFS model is used for weather prediction at ECMWF, but it is also used as the atmosphere component in EC-EARTH. As such, the community is much larger than ECMWF alone and it is used throughout Europe by many institutes and universities for weather and climate forecasts and research. The IFS model discretizes the 3D Navier-Stokes equations and it uses a spectral method to compute the dynamics of the atmosphere. Atmospheric physics, e.g. precipitation, is calculated on each grid-point, for which the domain is decomposed in two dimensions. Therefore, the solution needs be converted between spectral space to calculate the dynamics and grid-point space to calculate the physics at each grid-point. This involves two MPI_Alltoallv calls at every timestep. IFS uses the double precision General Matrix Multiply (DGEMM) routine from the BLAS library for part of its computations. IFS can be built as a hybrid application, with support for both MPI and OpenMP.

1.2. NEMO

NEMO is an ocean model that includes several components besides the ocean circulation, including sea-ice and biogeochemistry [2]. It discretizes the 3D Navier-Stokes equations and it uses MPI with a 2D domain-decomposition approach as paralellization mechanism. The core development team of NEMO consists of members of CNRS and Mercator-Ocean in France and UKMO and NERC in the United Kingdom. The NEMO ocean model is used in several climate models that are used for the IPCC reports about climate change (notably HadGEM3, EC-EARTH and IPSL-CM4). Furthermore, it is also used for standalone simulations.

Several computing centers are interested to work on improving the performance and/or scalability of NEMO: e.g. STFC is working on implementing a decomposition with variable domains. The IPSL team in Paris is involved in both the development of NEMO and the scientific use of NEMO and have been involved in PRACE1PP as well.

1.3. OASIS

The OASIS coupler, currently developed by CERFACS (France), DKRZ (Germany), and Centre National de la Recherche Scientifique (France) in the framework of the EU FP7 IS-ENES project is software allowing synchronized exchanges of coupling information between numerical codes representing different components of the climate system. Portability and flexibility are the key design concepts of OASIS3 developed and maintained since more than 15 years at CERFACS [3]. OASIS3 is currently used by approximately 30 climate modelling and operational weather forecasting groups in Europe, USA, Canada, Australia, India and China. Parallellism was later added to OASIS3 and it is implemented by dividing the coupling fields over the available OASIS3 processes. The

maximum parallellism in the coupled is therefore limited by the number of coupling fields.

1.4. Confidentiality

The EC-Earth model has a license via ECMWF that is not open source, but it allows for free 'Academic use' for users in member states of ECMWF (most European countries). The NEMO ocean model is open source (French CeCILL license) and as such there is no confidentiality. However, specific setups might be confidential, since research groups invested a lot of time into the preparation of files with initial- and boundary conditions. The OASIS3 coupler uses the LGPL open-source license.

1.5. Scientific cases

T159-ORCA1: the atmosphere model IFS is run with a 'resolution' of T159, which converts to a grid-spacing of approximately 128km. The ocean model NEMO is run at a 1 degree, which translates to about 110km. The resolution of the atmosphere and the ocean model match quite closely. However, since the IFS model incorporates a lot more physical processes, it requires several times more computing resources than the NEMO model. The atmosphere-to-ocean coupling involves 14 fields, while the ocean-to-atmosphere coupling involves 6 fields. The input data for the IFS model are about 29MB in size. The timestep is one hour, both in the atmosphere- and in the ocean model.

T255-ORCA1: EC-EARTH 3.1 was released in November 2011 and changed the default resolution of the atmosphere to T255, but otherwise it is the same as the above model. The input data for the IFS model are about 95MB in size.

T799-ORCA0.25: high-resolution version of EC-EARTH, with a resolution of about 25km and a similar resolution in the ocean. Just like the lower resolution version, the atmosphere-to-ocean coupling involves 14 fields, while the ocean-to-atmosphere coupling involves 6 fields. Note that the high-resolution version does not yet include coupling of runoff from the IFS atmosphere model to the NEMO ocean model. The input data for the IFS model are about 680MB in size. The timestep is of the atmosphere model is 12 minutes, the ocean model uses 20 minutes and the sea-ice model uses 1 hour.

Table 1: Parameters of different EC-EARTH configurations

Parameter	T159-ORCA1	T255-ORCA1	T799-ORCA0.25
IFS Timestep	1 hour	1 hour	12 minutes
IFS Input file size	29MB	95MB	680MB
IFS grid snapshot size	12MB	16.5MB	107MB
IFS grid snapshot # 2D slices	161	63	63
IFS spectral snapshot size	13MB	6MB	67MB
IFS spectral snapshot # 2D slices	250	45	45
IFS Snapshot frequency	6 hourly	6 hourly	6 hourly
NEMO timestep	1 hour	1 hour	20 minutes
NEMO output frequency	varying: daily, 5-daily, monthly, ..	varying: daily, 5-daily, monthly, ..	varying: daily, 5-daily, monthly, ..

LIM2 timestep	1 hour	1 hour	1 hour
OASIS coupling frequency	3 hours	3 hours	3 hours

The atmosphere and ocean field of all model configurations are coupled every 3 hours. IFS writes out snapshots every 6 hours. The NEMO uses a more sophisticated I/O system, with small output sets at daily frequency, and more significant output every 5 model days.

1.6. Goals

The following actions were defined at the start of this collaboration:

1. to make a performance analysis of the high-resolution configuration of the coupled EC-EARTH 3 on different platforms. Is the bottleneck I/O, communications or load-balancing in the coupler?
2. Validate the OASIS4 coupler for the EC-Earth coupling configuration.
3. Assess the performance of and improve the coupling implementation, including the potential benefits and costs of upgrading to OASIS4. If beneficial, upgrade EC-EARTH 3 to OASIS 4.
4. Mapping of MPMD hybrid MPI+OpenMP threads and MPI-only tasks efficiently onto cores on different architectures.
5. Investigate if CUDA-enabled routines can improve the scalability of the coupled or atmosphere-only model. This excludes rewriting routines into CUDA, which is part of PRACE 1IP Task 7.5.
6. Investigate and improve scalability of I/O within IFS and NEMO.
7. Design of one application to run ensemble simulations. Monitoring and fault tolerance of such a mega-model (which, in itself, is desirable) must be addressed in the design. The final implementation of such an ensemble system is of secondary importance and is only addressed if time permits.
8. To optimize the mapping of MPI tasks of the submodel NEMO (possibly also IFS) to cores on different architectures.
9. To implement dynamical memory allocation and load-balanced domain decomposition.
10. Look at scalability of reading of forcing file for ocean-only run.
11. Improve the efficiency of CDO for postprocessing of GRIB data, which is very I/O intensive. This is partly due to archiving, but also do to the I/O ‘heavy’ CDO.

The items 2, 3, 7 and 9 are not funded by PRACE. For the climate community, the goal is to be able to simulate 10 years of climate per wall clock day, which converts to 1 simulated hour per second (assuming every year has 360 days). As the resolution becomes higher, the timestep is shortened to fulfill the CFL-criterion. For the lowest resolution models that are analysed in this paper, the timestep is one hour for the atmosphere and the ocean, so the goal would be one timestep per wall-clock second. For the high-resolution T799 IFS submodel, the timestep is 12 minutes, so the goal is to reach 5 timesteps per second.

1.7. Test systems

Most tests have been done on the PRACE Tier-0 system Curie. The hardware and the software of the Curie system are given in Table 2.

Table 2: Hardware and software of used systems

System	Curie	Huygens	Ekman
Processor	Intel(R) Xeon(R) CPU X7560 @ 2.27GHz	IBM Power6 @ 4.7GHz	AMD Opteron 2374HE
Interconnect	InfiniBand QDR Full Fat Tree network	InfiniBand 8x DDR fully connected network	InfiniBand 4x DDR, multiple root tree network
Node	32 cores	32 cores	8 cores
global file-system	Lustre	GPFS	Lustre
OS	Bull linux	Suse linux ES11	CentOS Linux 5
Compiler	Intel compiler	IBM XL compiler	
MPI library	bullxmpi	IBM POE	

Initial tests were run on the Ekman system in Sweden and the Huygens system in the Netherlands.

1.8. Tools used

Developed at the Jülich Supercomputing Centre, Scalasca is a performance analysis toolset that has been specifically designed for use on large-scale systems including IBM Blue Gene and Cray XT, but also suitable for smaller HPC platforms using MPI and OpenMP [4]. It determines the computation and communication time of each subroutine and for each MPI task and presents the results in a convenient GUI. Furthermore, it gives details like the number of communications and the total number of bytes that was sent and received. This can be used to optimize the settings for the MPI-library. Although by default it gives a time-averaged view of the application, the source code can be instrumented manually to separate different parts of the code at runtime, e.g. to filter initialization and finalization periods that may distort the results for short benchmark runs.

Darshan [5] has been used to diagnose the I/O of a short simulation of 2 days with EC-EARTH. This tool analyzes all I/O operations and writes this to file at the end of the run. A summary can be generated from the darshan output that gives an overview of all the I/O operations, e.g. the times that files are opened, and the amount of data that is read and written.

2. Results

2.1. Porting

The EC-EARTH model consists of three separate executables, several input scripts and a large submit script. Release 3 of the EC-EARTH model was in heavy development during the whole period of the PRACE task. Several new versions of the EC-EARTH model were released over the duration of this project, each with significant improvements in portability. On the other hand, porting a new release would always take at least a week to get running again, which is an indication for the complexity of the EC-EARTH model.

SMHI (the Swedisch Meteorological Institute) was leading the development and was very helpful to get the model ported and running on different architectures. The collaboration between different developers throughout Europe was organised efficiently through a website that was used to host the software, share plans and have discussions about errors, etc. Bug reports were always answered quickly and the main developers were very helpful.

The main system for our developments, was the Curie system, but this was unfortunately not available until spring 2011. Until that time, developments were done on local systems (mainly Huygens and Ekman), all with different software and performance characteristics. Progress was therefore slow during the initial period.

E.g., porting EC-EARTH to the Huygens system at SARA, the most important issue was the mix of REAL*8 and DOUBLE PRECISION, which is not always the same: in case the default REAL size is 8 bytes, then the size of a DOUBLE PRECISION number is 16 bytes on an IBM Power platform, but not so on an Intel-based platform.

Porting to the Curie system was relatively straightforward, probably helped by the fact that the used software and hardware on Curie resembles the system that is used for the main development. On Curie, the MKL library can be used to provide high-performance BLAS routines. When IFS is compiled with support for OpenMP, it is best to use the multi-threaded version of the MKL-library as well.

2.2. Timestep analysis

The IFS model consists of different modules, of which some are only run at certain intervals. The most important types of timesteps for our simulations are:

- computational timesteps: calculation of atmospheric dynamics and physics without radiation.
- radiation timesteps: calculation of atmospheric dynamics and physics including radiation. The frequency of radiation timesteps is hourly for resolutions above T511, otherwise every 3 hours.
- coupling timesteps: coupling of several fields between the atmosphere and the ocean. The timesteps with fields received from the ocean are denoted as coupling timesteps. At the end of the timestep before each coupling timestep, the atmospheric fields are sent to the ocean. The frequency is every 3 hours.
- I/O timesteps: every 6 hours.
- initialization and finalization timesteps

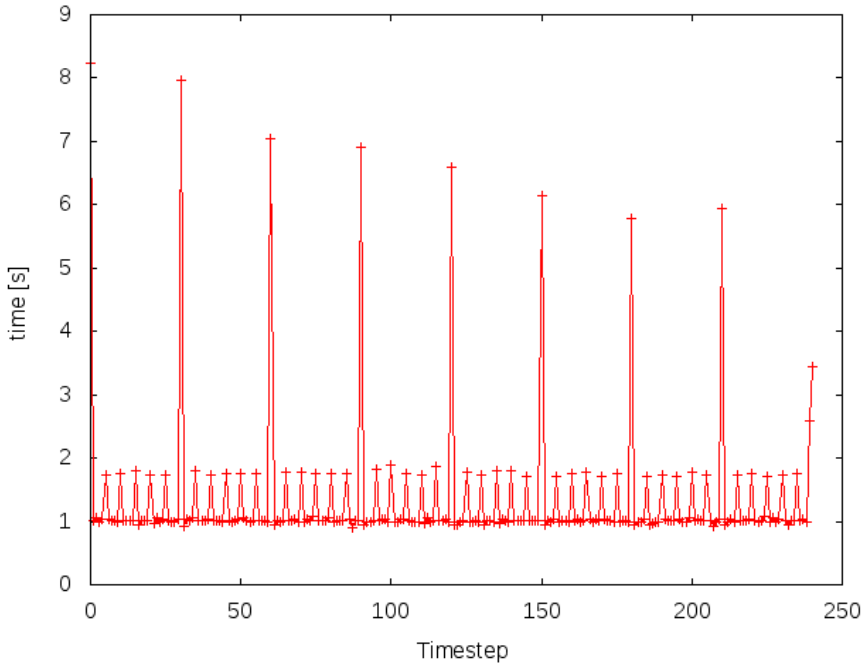

Figure 1: Time per IFS timestep for the T799-ORCA0.25 configuration on Curie using 10 OASIS tasks, 320 NEMO tasks and 1536 IFS tasks.

Note that coupling and I/O timesteps include full radiation timesteps.

Figure 1 shows the time for each consecutive timestep for a simulation of EC-EARTH on Curie at a resolution of T799. It clearly shows the impact of writing a snapshot every 30 timesteps. The I/O timesteps are quite variable in time, with fluctuations of 2 seconds (about 30%). Furthermore, the first and last timestep take quite long. This includes a lot of initialization and finalization routines, which are not important for the model performance of long simulations. However, when profiling the application, these timesteps need to be filtered out. The fastest timesteps calculate the dynamics and most of the physics of the atmosphere. The radiation is run hourly, which gives smaller peaks every 5th timestep. Coupling takes place every 15th timestep, but this is not noticeable in the figure. However, it shows that the time IFS takes between two coupling timesteps varies: one period includes an I/O timestep, while the next one does not. This has implications for the coupling efficiency: assuming that the NEMO model is adjusted to keep up with the IFS model at all times, it'll be waiting for the IFS I/O to complete.

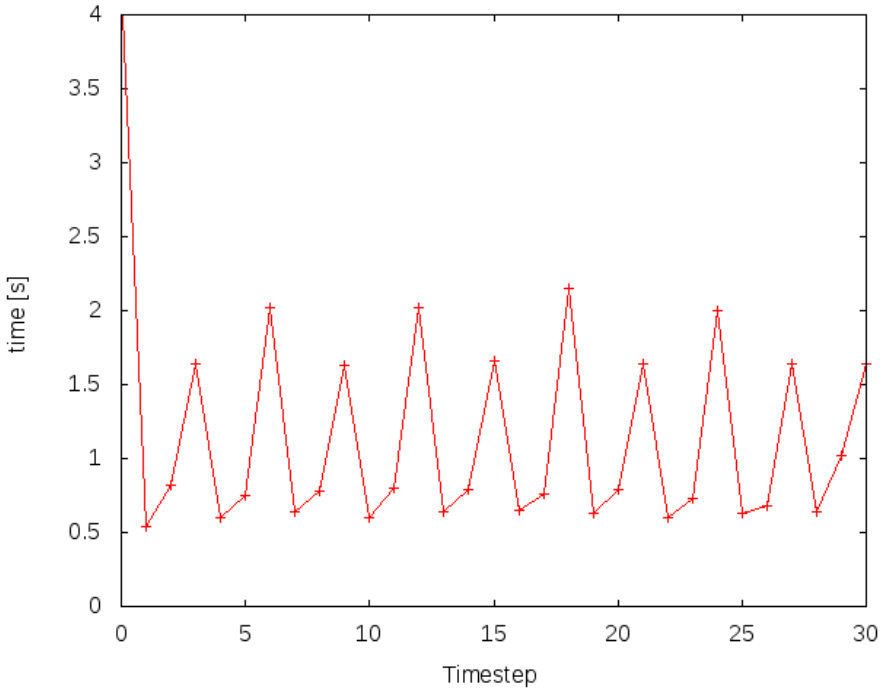

Figure 2: Time per IFS timestep for the T159-ORCA1 configuration on Curie using 8 OASIS tasks, 64 NEMO tasks and 96 IFS tasks.

Figure 2 shows the time for each consecutive timestep for a simulation of EC-EARTH on Curie at a resolution of T159. Since the length of a timestep is one hour, the coupling and I/O occur more often. Every 3rd timestep shows the impact of the radiation calculation and the coupling to the ocean, while every 6th timestep shows the added impact of writing a snapshot. More interestingly, the timestep before coupling is also longer, which is due to the atmospheric runoff field being processed and sent to the ocean. The algorithm and how it can be scaled up, is explained later.

Table 3: Fraction of all timesteps for each type.

Timestep type	T159-ORCA1	T255-ORCA1	T799-ORCA0.25
Computational	66,60%	66,60%	80,00%
+Radiation	0,00%	0,00%	13,34%
+Coupling	16,70%	16,70%	3,33%
+I/O	16,70%	16,70%	3,33%

The EC-EARTH output contains timings for the IFS timesteps, which have been analysed with an awk-script that can be found in Appendix A.

The performance of the fully-coupled model is determined by the performance of the slowest submodel. Therefore, different submodels need to be balanced in order to achieve the best performance. As was shown in the previous section, the NEMO model is several times faster than the IFS model for the same number of MPI tasks, due to the more complex physical processes that are simulated in the IFS model. NEMO can therefore be run with more MPI processes than is strictly necessary to keep up with IFS, since NEMO uses a small part of the resources anyway.

The following paragraphs further analyse the different types of timesteps for the T799 model on Curie. For simplicity, the EC-EARTH model was run with a fixed setup for NEMO and OASIS. NEMO uses 320 MPI tasks and OASIS uses 10 MPI tasks. It was deduced from scalasca results that the NEMO model with 320 MPI tasks is

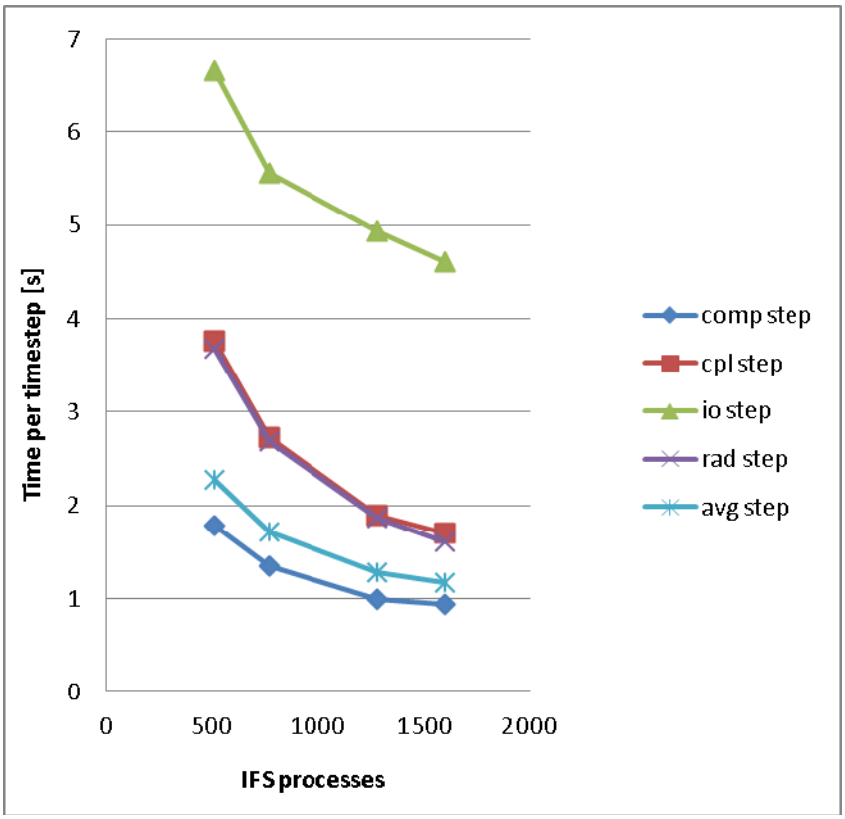

Figure 3: Time needed for different types of timesteps as a function of the number of IFS tasks. The nr. of NEMO and OASIS processes is kept constant at 320 and 10, respectively. The weighted average timestep (weights can be found in table 3) is also added.

almost twice as fast than IFS with 1024 MPI tasks.

Figure 3 shows the time per type of timestep, figure 4 shows the speedup per type of timestep and figure 5 shows the total cpu time needed for each type of timestep. At first glance, I/O seems to be the bottleneck, with the highest time per timestep and the poorest scaling properties. However, when we look at the fraction of the total CPU time that is spent for I/O, this is still minimal. I/O is expected to be a bottleneck when the application can be scaled to higher core counts, or if the frequency of I/O timesteps increases.

Coupling was thought to be a bottleneck as well, since it uses a limited number of separate binaries to communicate the coupling fields between the atmosphere and the ocean. Surprisingly, all of the figures show that this is not the case: scalability is good and the total CPU time spent on coupling is small. Remember that coupling timesteps include a full radiation timestep, so the overhead is minimal as can be seen in figure 3. This may change when the application is scaled to higher core counts, or if the frequency of coupling timestep increases. However, the OASIS team has developed a more scalable solution at the end of 2011, OASIS3-MCT, which is shown to scale to at least 2000 cores with less than 1% overhead and is expected to keep minimal overhead for larger core counts. Although this was implemented in the latest release of EC-EARTH, this was not yet ported to Curie. This work is still ongoing.

The latest EC-EARTH release includes the NEMO 3.3.1 model. The scalability of the NEMO component was increased significantly in version 3.4, released in February 2012 [6]. The NEMO model used a MPI_AllGather call across a small subset of all its tasks, which was shown to dominate performance at scale. This was replaced by a limited amount of MPI_Sends and MPI_Recvs, similar to the exchange of halos. This basic idea for the improved scalability was developed and implemented during the PRACE Preparatory Phase project, but was implemented in the official release by Andrew Coward too.

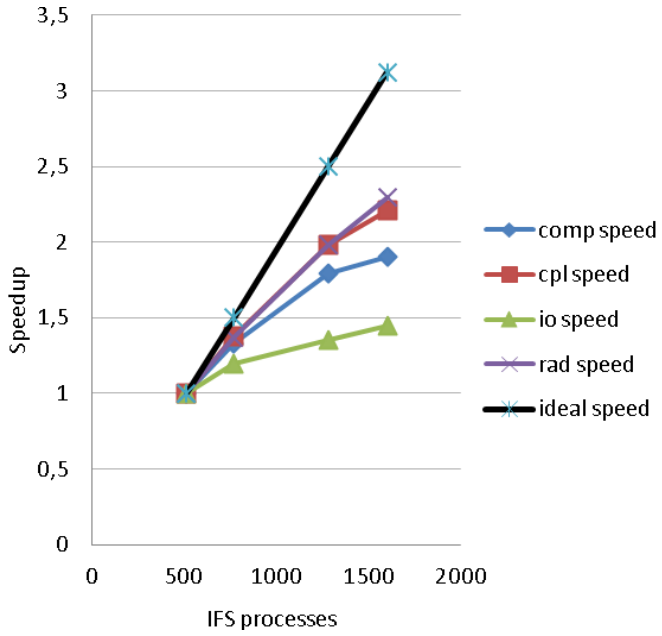

Figure 5: Speedup for different types of timesteps (w.r.t. 512 IFS tasks) as a function of the number of IFS tasks. The number of NEMO and OASIS processes is kept constant at 320 and 10, respectively.

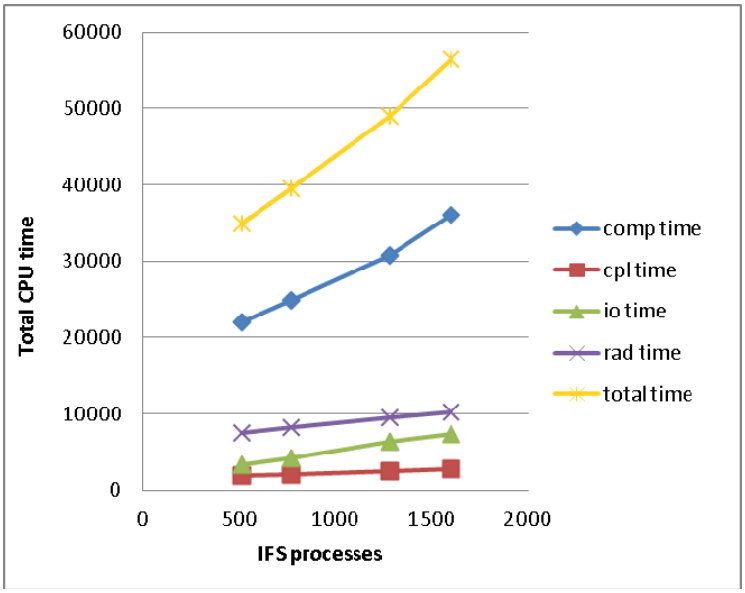

Figure 4: Total cpu time needed for different types of timesteps as a function of the total number of MPI processes. The number of NEMO and OASIS processes is kept constant at 320 and 10, respectively. The total cpu time is the sum of all types of timesteps. A horizontal line would indicate perfect scalability. Lower is better.

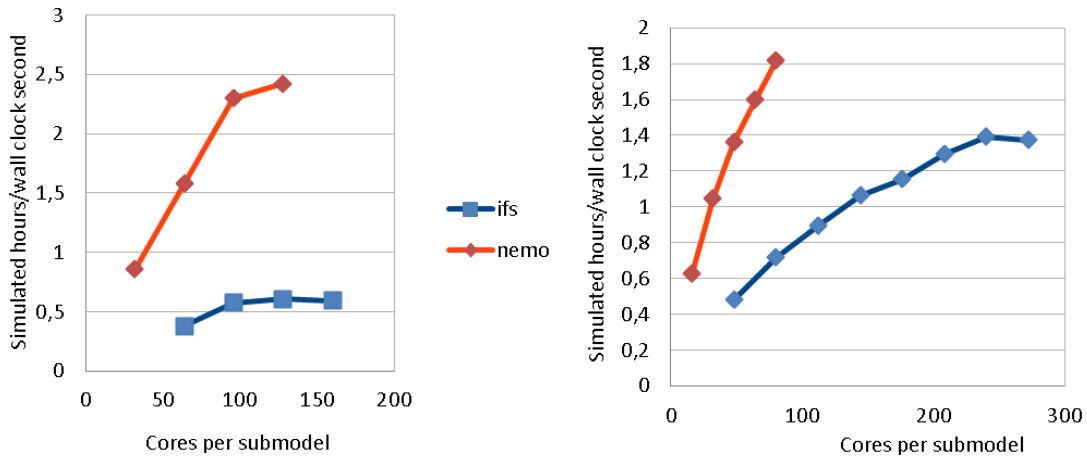

Figure 6: Scalability of IFS and NEMO submodels at T159-ORCA1 configuration as a function of the number of processors per submodel in simulated hours per wall clock second on (a) Curie and (b) Huygens.

At 1600 cores, the largest part of the increase in total CPU time comes from the computational timesteps (see figure 5), primarily since these make up 80% of all the timesteps (see table 3). Although the computational timesteps are the fastest, their scalability is not optimal.

2.3. Performance analysis with Scalasca

Scalasca was used to determine the scalability of the atmosphere- and ocean components independently. This can then be used to find a balanced combination of atmosphere- and ocean-processes, to minimize the waiting time when coupling both models. As was noted before, both models are never equally fast, due to different modules (most notably I/O) that are not called at every coupling period. As the required number of processors for IFS is much larger than for NEMO, it is best to let NEMO wait for IFS to finish its I/O. Furthermore, a scalasca analysis can give insight as to what component might be the primary bottleneck for the performance of the coupled model.

Scalasca normally gives a time-averaged overview of the performance of an application, but with some manual instrumentation it is possible to e.g. filter out the initial timestep, as this is not representative for the main part of a production run. The main loop was splitted into three different timesteps: timesteps with only computations, with coupling and timesteps with I/O. The source code changes can be found in Appendix B.

Figure 6 shows the performance of both submodels as a function of the number of processes for the Curie and Huygens systems. Ideally, to select a balanced combination of submodels, one would select the required performance on the vertical axis and read off the required number of processes for IFS and NEMO. The goal for climate scientists is to reach one simulated hour per wall clock second, which enables long term climate simulations. It is clear that the scalability and performance of NEMO is enough to reach that goal, even at low core counts, with a similar performance per core on Curie and Huygens. IFS unfortunately does not reach the required performance on Curie and the scaling stops well before. This is mainly due to the poor performance of MPI calls (both Alltoallv and Isend+Irecv) with the default options for the MPI-library on Curie.

Scalasca output shows that almost the complete increase in CPU time in IFS is due to increased communication time, primarily the two calls to MPI_Alltoallv at each timestep (see Table 4).

Table 4: Computation and communication times for EC-EARTH T799-ORCA.25 on Curie

	512 IFS processes	1024 IFS processes	2000 IFS processes
Total (kCPU s)	303	361	
Execution (kCPU s)	211	214	230
Communication (kCPU s)	86	135	324
Computation per process (s/ts)	1.7s	0.87s	0.48s
Communication per process (s/ts)	0.7s	0.55s	0.675
MPI_Alltoallv (kCPU s)	57	92	216
MPI_Wait (kCPU s) waiting for MPI_Isend and MPI_Irecv	22	32	

The two MPI_Alltoallv calls transfer in total 2.5+1.25GB per timestep. It is clear that the computational part of the code is very scalable, with only a 1% increase when using 1024 instead of 512 IFS processes. Unfortunately, communication time per task is clearly becoming the bottleneck at 1024 IFS processes. It was shown in another PRACE report that the MPI_Alltoallv call on Curie is by default quite slow and can be optimized dramatically, at least 5 fold, when using 1024 cores [7]. Work is ongoing on the implementation of an efficient MPI_Alltoallv call for Curie in IFS.

The benchmark program in appendix c can be used to test the MPI_Alltoall performance. It sends 2.5GB of data around, independent of the number of MPI tasks. The size of an individual message therefore decreases quadratically with the number of MPI tasks. It is therefore a good test for the strong scaling potential of IFS at T799 on a system. When using the default settings, the MPI_Alltoall call takes almost 0.2s.

The BullxMPI library can be tuned for the Alltoallv call with the following settings:

```
export OMPI_MCA_coll_tuned_use_dynamic_rules=1
export OMPI_MCA_coll=tuned,basic
export OMPI_MCA_coll_tuned_alltoallv_algorithm=2
```

which results in a performance improvement of 25% on Curie for the benchmark program at 1024 MPI tasks. Unfortunately, we have not yet seen an improvement for EC-EARTH with the same settings. Table 5 shows that the use of OpenMP could significantly speedup the communication, even if only using two OpenMP threads. A notable exception is the timing of “MPI-only” on 4096 cores with default options, for which we do not have an explanation.

Table 5: Timing of MPI_Alltoall on different numbers of cores, showing the effect of OpenMP on communication. The first number is measured when using default BullXMPI options and the second number uses the option tuned_alltoall_algorithm=2.

	1024 cores	2048 cores	4096 cores
1 “thread” (“MPI-only”)	0.19 – 0.14	0.24 – 0.13	0.075 – 0.75
2 “threads”	0.073 – 0.070	0.14 – 0.075	0.17 – 0.099
4 “threads”	0.072 – 0.073	0.070 – 0.051	0.15 – 0.065
8 “threads”	0.074 – 0.074	0.055 – 0.056	0.031 – 0.034

2.4. Profile of IFS

Almost all cpu-time in the IFS-model is spent in the forward integration of the climate model. This is coordinated from the CNT4 routine in the IFS-model. The most important subroutines of a timestep in this routine (in order of appearance in the code) are:

- (during coupling timesteps) GETOASIS3: get fluxes from the ocean (wrongly indicated as 'send fluxes to..') [UPDTIM-> UPDCLIE-> GETOASIS3]
- (during io timesteps) GRIDFPOS+DYNFPOS: write grid-point output file and spectral output file
- TRANSINVH:
    - LTINV: Direct Legendre transform using DGEMM routine in LEINV
    - TRMTOL: Tanspose data from wave number partitioning to latitude partitioning using MPI_Alltoallv (TRMTOL)
    - FTINV: Transform from fourier-space to grid point-space using built-in FFT routine.
    - TRLTOG: Transpose of grid-point data from latitudinal to column. Many Isends and Irecvs.
- GP_MODEL: Physics routines
- TRANSDIRH:
    - TRGTOL: Transpose of grid-point data from column to latitudinal. Many Isends and Irecvs.
    - FTDIR: Transform from grid-point to fourier-space using built-in FFT routine.
    - TRLTOM: Transpose data from latitude partitioning to wave number partitioning using MPI_Alltoallv.
    - LTDIR: Direct Legendre transform using DGEMM routine in LEDIR.
- SPCH:
    - TRMTOS/TRSTOM: transposition between horizontal and vertical spectral coefficients and vice versa. Many ISends and Irecvs.
    - SPCSI: Spectral space semi-implit computations.

- (only during timestep before a coupling timestep) PUTOASIS3: send fluxes to the ocean. If runoff is coupled, then the routine RUNOFF_TO_OCEAN first gathers a field onto the first processor. The first processor sums the gridded field into 66 basins, which is then redistributed over grid points that are associated with each basin. The new gridded field is redistributed across all processors. The gathering and scattering of the gridded fields (which can be several MB's) can be replaced with an MPI_Allreduce call of one floating-point value for each of the 66 basins, which reduces the size of communications tremendously. Also the computational part would then be distributed. At the time of the performance analysis, this was only implemented for the T159 model, although this is used at higher resolutions as well.

Many subroutines that do not take a significant amount of time are left out and several subroutines are called at a much lower level than directly from CNT4.

2.5. Scaling OASIS coupler

The OASIS3 coupler uses separate MPI processes that send and receive messages with coupling fields between the different components. Although OASIS3 is serial for a single coupling field, it can use different processes to couple multiple fields in parallel (this is called 'pseudo-parallel'). Due to the EC-EARTH coupling sequence (IFS and NEMO run in parallel), a large amount of the time needed for coupling (MPI message exchanges + interpolations) is hidden by the speed difference between the two components: most of the coupling operations are performed when the fastest model has ended its computations and before the slowest model has ended theirs. As was mentioned before, since I/O timesteps do not take place in between all coupling timesteps, it is impossible to find a perfect balance between both models. OASIS4 was meant to replace OASIS3 with a fully parallel algorithm and many more features like unstructured, adaptive grids. Unfortunately, this could not be implemented on time. An alternative approach was used, with the OASIS3 interface and the 'Model coupling toolkit' (MCT) as the backend, called OASIS3-MCT. The advantage is that this is fully parallel and works as a library that is linked with the main components. Although it does not provide all features that OASIS4 would provide, it is enough to couple EC-EARTH 3.1.

Started on the ekman machine, the replacement process consists in a few operations:

- into code interfaces (a single mod_prism module has to be called instead of a suite of specialized modules)
- in the namelist file namcouple (OASIS3-MCT is currently not able to calculate interpolation weight, but reads it frp, a file, which name must now be specified in namcouple)

Several FORTRAN-related inaccuracies (argument array dimensions) have been corrected in the IFS and NEMO coupling interface to be able to reproduce, with OASIS3-MCT, the identical coupling process of OASIS3, including coupling field restart read/write.

Benchmarks with a different coupled climate model show a benefit of 18% at 1000 computing tasks on ekman [9]. Note that a large part of this benefit will be hidden in EC-EARTH, because both components are run in parallel and are not exactly equally fast.

2.6. Implement standard FFT calls for possible use of GPGPUs to calculate FFTs

The IFS model now uses a built-in routine FFT992 to calculate FFTs. To enable the use of GPUs for the calculation of FFTs, the FFT992 functions were replaced with an implementation to use the standard FFTW3 interface for 1-D FFT calls. IFS can now use standard external FFT libraries. This was tested for correctness on the T159L52 test cases on the stokes cluster at ICHEC using the Intel MKL implementation of FFTW3. Furthermore, it allows the use of GPGPU hardware through the CUDAFFT-library. Testing is underway to find the performance gain for IFS of using GPGPU hardware.

2.7. OpenMP in IFS

The IFS model is a hybrid MPI and OpenMP application, although only MPI has been used so far. The OpenMP-version of IFS often creates problems due to e.g. library routines that are not thread-safe. However, the potential impact of the use of OpenMP for the scalability of IFS has been shown with the MPI_Alltoall benchmark program from appendix c: when running the benchmark with 16 MPI tasks per node, simulating the use of two OpenMP threads, the time decreases by more than half. This would directly translate in a more scalable IFS model.

Compilation of IFS with OpenMP on Curie is straightforward. Adding the Intel compiler option `-openmp` is all that it takes. Part of the direct Legendre transform uses the BLAS routine DGEMM, for which it is best to use the optimized vendor-libraries. On the Curie platform, this is the MKL library from Intel. MKL also provides multi-threaded versions of these libraries, which need to be used when OpenMP is enabled. Also, the implementation of the standard FFTW3-call as described in section 2.62.6 allows the use of a multi-threaded FFT-library.

To run the model, the following two environment variables need to be set:

```
export OMP_NUM_THREADS=2
export OMP_STACKSIZE=1G
```

As a first test, the EC-EARTH model was run with two OpenMP threads and two reserved cores per MPI process. Unfortunately, the resulting hybrid executable is not faster than the MPI-only executable.

Scalasca was tried to find the reason why OpenMP would not increase performance. Scalasca supports OpenMP through the ompP profiler, but the parser was not yet fully compliant during 2011. The compilation needed to be done serially, since the ompP parser used temporary files in each directory, which are updated at every compilation step. Compiling IFS with OpenMP and Scalasca therefore took many hours. Unfortunately, IFS could not be compiled with Scalasca and OpenMP, because of the following error when compiling src/ifs/ald_inc/interface rtl_setup.F90:

```
terminate called after throwing an instance of 'std::out_of_range'
what(): basic_string::substr
/sara/sw/scalasca/1.3.3/bin/kinst: line 499: 19408 Aborted
/sara/sw/scalasca/1.3.3/bin/opari -disable region rtl_setup.F90
```

This error was due to the combination of temporary files and the source file, since the source file can be compiled and instrumented by scalasca without temporary ompP files.

At the end of 2011 the scalasca team released version 1.4, with vastly improved support for OpenMP, including parallel compilation. The compilation of EC-EARTH would get further than before, but still several parsing issues remained. These parsing issues were communicated with the developers and these were partially fixed in version 1.4.1. It is expected that the remaining issues will be fixed in a future release of scalasca.

2.8. Process distribution

The Curie launcher by default distributes the MPI processes across the nodes in a blocked fashion. The first node is filled with MPI processes and then the next node. This is usually a good choice, because neighbouring MPI processes are usually also neighbours in the simulation and require therefore the most communication. However, for EC-EARTH it may be beneficial to distribute the processes cyclically across the nodes. The first 10 processes are used for the OASIS coupler, each of which will send and receive messages to all processes when fields are coupled between the atmosphere and the ocean. Especially at scale, this large amount of messages may cause a queue in the communications. Also, tests with the stand-alone NEMO model show that the cyclic distribution is beneficial [10]. As these experiments were conducted and documented, with explanation, this was not repeated as part of the PRACE collaboration.

To test if a cyclic distribution of processes would improve the performance of EC-EARTH, an experiment was done using the T799-ORCA0.25 configuration with 1600 IFS processes, 320 NEMO processes and 10 OASIS processes. On Curie, a node cyclic distribution can be set as follows:

```
ccc_mprun -E '-m cyclic' ./a.out
```

There was no discernible difference in performance between using a block and a cyclic distribution. A similar distribution for a low-resolution configuration on Huygens showed a small performance penalty. This might be due to variations in timing from run to run, but it shows at least that no large gain is to be expected from a cyclic distribution.

Since the OpenMP-version of EC-EARTH on Curie was not faster than the MPI-only version, it was not investigated how to distribute a mix of MPI-only and hybrid applications efficiently across nodes on different systems.

2.9. I/O analysis

As was shown in previous sections, I/O is not a bottleneck for EC-EARTH at the highest resolution and for the current settings, but this could change when the scalability of the computational part is further increased, or the frequency and amount of output is increased. Therefore, we have added a section on I/O, that describes the I/O system in general and gives some hints as to what could be done about it.

Three I/O bottlenecks for highly parallel applications are:

1. large amounts of data that need to be read or written. Both the IFS and NEMO models use serial I/O, so the data is gathered on a specific I/O rank and written to disk. When reading files, data is read from disk on a specific I/O rank and broadcasted or distributed across all MPI ranks. NEMO can also write files in parallel, where each MPI task writes its own part of the output to a separate file.
2. large amounts of I/O operations, e.g. opening and closing files.
3. post-processing of data. The first bottleneck can usually only be solved by parallellising the I/O, either through writing multiple files or using parallel I/O. Unfortunately, the IFS model uses the GRIB file format, which is not well-suited to parallellisation.

IFS has quite rudimentary I/O options, where it is only possible to write snapshots. The I/O routines are implemented in the module IOSTREAM_MIX. The default configuration writes out a snapshot every 6 hours. Every snapshot generates two files, one is on a spectral grid (written by WROUTSPGB) and another is on a regular lat-lon grid (written by WROUTGPGB). The domain is splitted horizontally across MPI processes, while the fields are splitted vertically in the GRIB output files. Therefore, each file is written to in two steps in the routines GRID_OUT and SPEC_OUT:

1. Routine GATH_GRID and GATH_SPEC: all the data is redistributed using MPI_Alltoallv, so afterwards each process holds one or more fields that are vertically separate slices, but with the full horizontal domain. These fields can then be further processed.
2. Routine WRITE_RECORD: the fields are sent to the first IFS task and written to disk.

On Curie, most time is spent in MPI communication to redistribute and gather the fields, not in the actual writing of the files. Furthermore, for each IFS process about 100kb of data per snapshot is written to disk. This is relatively constant with model resolution, because the number of cores is increased with resolution.

To remove the communication bottleneck, each MPI task would need to write its own data to disk. This would result in many, small files which is usually not preferred for postprocessing. Also, the number of file operations would increase, which might put extra pressure on the parallel file system. Parallel I/O can be used to keep the I/O local to each process (or a small group of processes). To increase the data size per I/O-operation, data should be kept in memory for multiple snapshots.

With a buffer of 10MB per task, EC-EARTH can keep about 100 snapshots in memory for delayed writing. This would mean one I/O operation of several tens of GB per simulated month. Since the used file format does not allow easily for parallel I/O, this would only be possible if the file format is changed. Since the EC-EARTH community uses the NetCDF-format for their analyses and post processing and there are also two different parallel implementations, this would be the recommended format to use.

The PIO library for parallel I/O of earth system models [12] reaches about 1GB/s. Performance tests on Curie [11] have shown that a write speed of about 0.5GB/s at 2048 cores can be reached. Assuming that EC-EARTH would run at the preferred speed of one simulated hour per second, the I/O would take 5-10% of the runtime. The same report also shows an important caveat: the performance depends a lot on the internal structure of the generated NetCDF (or HDF5) file and it could be lower than 0.1GB/s. It is therefore recommended to design the NetCDF-output files and test its performance before actually implementing this strategy in IFS. Furthermore, this first needs to be synchronized with the EC-EARTH community before such a large project can be undertaken. Unfortunately, there was not enough time during this collaboration to change the I/O in IFS.

A new I/O server (XIOS) for NEMO was under development for most of the PRACE1IP period, but unfortunately we have not been able to contribute to its further development. An investigation of the scalability of reading forcing files was therefore also put off. The first release of the I/O server came too late to meet the deadline for the PRACE 1IP task 7.6 white papers.

The second bottleneck is often a cause of unnecessarily reading configuration files or writing log files. Although this can be useful and harmless when the model is run with a small number of MPI processes, this can slow down the model considerably when thousands of MPI tasks access the same set of files on the file system continuously. EC-EARTH involves three binaries, each with its own set of start dumps, input files, name lists and logs. A later paragraph describes the options for the different submodels to minimize these I/O-operations.

Figure 7 shows the disk activity during the EC-EARTH run as measured with the Darshan tool, with the caveat that the figure only shows the begin and end of activity for each file, not the intensity within that period. The reading activity shows clearly the differences between the 3 binaries: the first 4 ranks for the OASIS coupler, then 40 ranks for the IFS model and finally 20 ranks for the NEMO model.

The OASIS coupler should not have very little I/O operations after the initialization, also because the verbosity and logging options have been disabled or minimized. However, the OASIS coupler shows continuous reading throughout the simulation (not only inferred from the figure), due to erroneously reading the same input data at each time step. The OASIS developers noticed this as well (actually before this analysis) and have patched OASIS to remove this unnecessary I/O.

The IFS model only shows reading activity during the initialization, but every task writes to its own logfile throughout the run. This can be resolved by disabling logging, as explained in the next section.

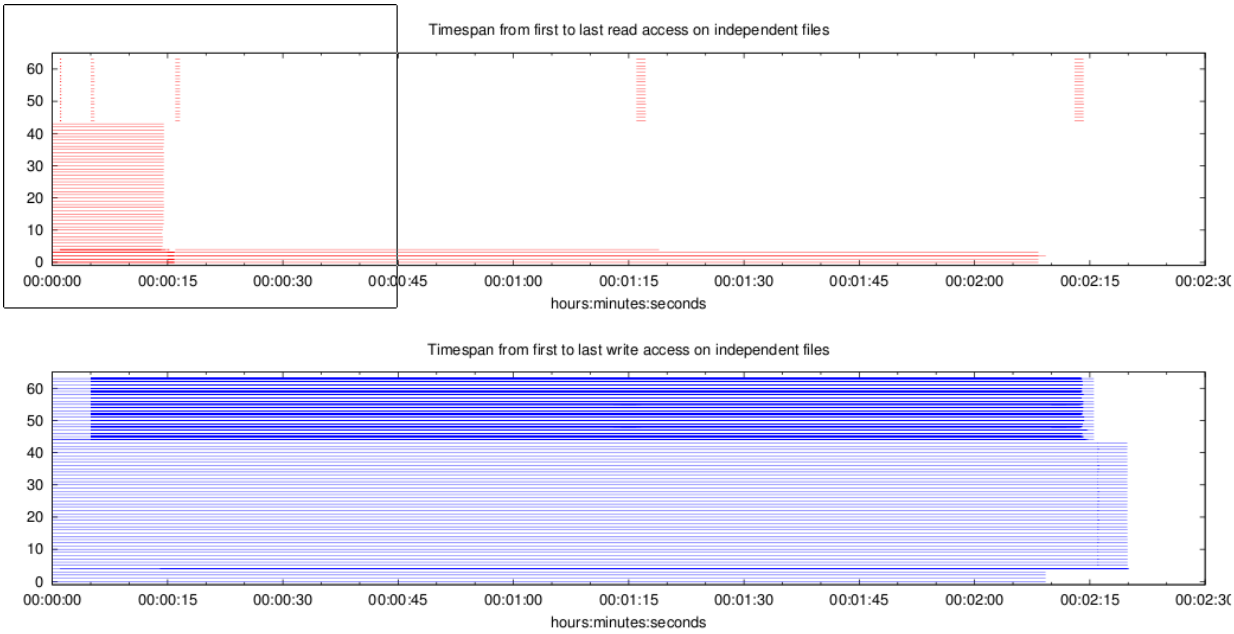

Figure 7: Timeline of I/O in a run of EC-EARTH on 64 cores. Top figure shows reading activity. Bottom figure shows writing activity. Note that a continuous line is drawn from the first read/write to the last read/write, but this does not indicate continuous activit

The NEMO model has some intermediate reads, which is most probably reading the files that describe the daily forcing. It is clear that all tasks are writing during almost the full run. It is known that the NEMO model writes some diagnostics about the solver and the completed timesteps to two files after each timestep, although this should only happen on the first task.

OASIS shows the following activity in this short run:

- 3500 writes to cplout_[MPI rank], throughout the run. The file is only opened once, and when NLOGPRT is lower than 2, the stream is not FLUSHed after each check.
- 400 reads of namcouple_[MPI rank], only at the very start of the run.
- 115 reads of rmp_A080_to_O1t0_GAUSWGT_0.nc, opened 80 times throughout the run.
- 45 writes to ifsmod.prt[MPI_RANK] by all OASIS ranks, at the initialization and the finalization of the run, but not during the run.

IFS shows the following activity:

- 22.234 reads of fort.4 by all IFS ranks. The file is opened twice and is read only during initialization of the run.
- 8.264 writes to NODE.001_01 by the first IFS rank, throughout the run. This can be switched off by setting NOUTPUT to 0 in the namelist file.
- 69 writes to ifs.stat, 9 writes to ncf927 by the first rank throughout the run.

- writes of dumps every 6 simulation hours by the first IFS rank. Output frequency is set with parameters NFRPOS and NFRHIS in IFS namelist. In cnt4.F the flags LLPLPP and LLPMLPP are TRUE in a step that should output data.

All submodels, including OASIS write out logging files and have several options that ease debugging. However, these produce quite a lot of output and I/O operations during the simulation. This can slow down the model and cause noise for the file system. It is therefore suggested to reduce these options as much as possible during production runs. The most important options are:

IFS:

- in namelist.ifs.sh, set NOUTPUT to 0. This disables writing of the NODE_001.01 file by the first MPI task: it opens /dev/null in ifs/setup/sumpout.F90 to write the log to.
- set OASIS3DEBUGLEVEL to 0 in the submit script. OASIS3DEBUGLEVEL is an environment variable used by IFS to change verbosity of IFS coupling routines.

OASIS:

- in namcouple, replace EXPOUT by EXPORTED. This ensures that fields are not written to disk when coupled.
- in namcouple.sh set NLOGPRT to 0. Disable verbose logging and multiple FLUSH commands during the run.

The third bottleneck is becoming more and more of a problem, with models scaling to higher resolutions with thousands of cores, while many of the postprocessing utilities are still essentially serial. The CDO tool (https://code.zmaw.de/projects/cdo) is used for converting of GRIB output files from IFS to NetCDF files that can be submitted to the CMIP5 project, which is a basis for the IPCC reports. This tool was found to be a bottleneck for processing large amounts of files, taking several hours to process one year of data on one system. This behaviour could not be reproduced on other systems and one of the important factors was the optimized compilation of CDO. Using the Intel compiler, it was found that the exact version was critical to reach a good performance. Moving from version 11.0.069 to 11.0.081 led to a ten-fold speedup on CDO on GRIB operations. Compiler optimizations, most notably SSE options and multi-thread support, will go a long way on the very regular operations in CDO. At an EC-EARTH meeting in Copenhagen in 2011 it was confirmed that CDO was not any longer the bottleneck in post-processing the data.

2.10. Framework to run ensemble simulations

A typical climate forecast experiment is a run of a climate model having variable range of forecast length from a few months to a few years. An experiment may have one or more than one start-dates and every start-date may comprise of single or many members. The full length of forecasting period for the experiment could be divided into number of chunks of fixed forecast length by exploiting model restarts. Furthermore, in the context of computing operations, every chunk could have two big sections; a parallel section where the actually model run would be performed and serial sections for performing other necessary operations like post-processing of the model output, archiving the model output and cleaning the disk space for the smooth proceeding of the experiment.

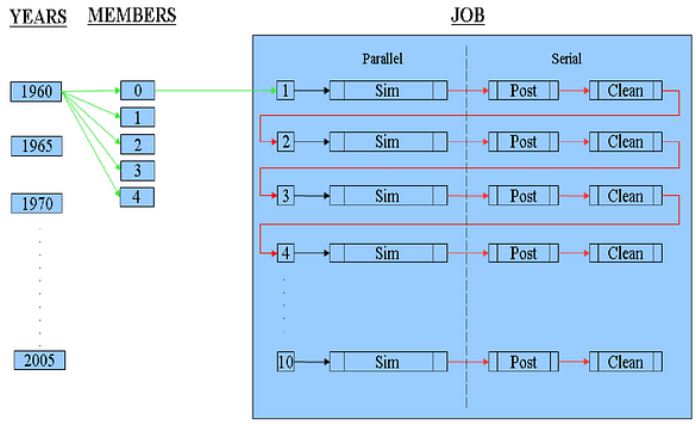

Figure 7: Sample experiment setup

Figure 8 shows a sample experiment which consists of 10 start-dates from 1960 to 2005 where every start-date is independent of each other and starting after every 5 years while each start-date comprise of 5 members. Every member is also independent and has been divided into 10 chunks which are dependent on each other. Now let us suppose that the forecast length for each chunk is one year and every chunk comprises of three types of jobs; a simulation (Sim), a post-processing (Post) and an archiving and cleaning job (Clean). Therefore with this typical example experiment, one start-date with one member comprise of 30 jobs and eventually 1500 jobs will be run in total for the completion of the experiment. In short, there is a need of a system to automate such type of typical experiments and optimize the use of resources.

IC^3 developed Autosubmit, which is a tool to manage and monitor climate forecasting experiments by using supercomputers remotely. It is designed with the following goals:

- efficient handling of highly dependent jobs;
- optimum utilization of available computing resources;
- ease of starting, stopping and live monitoring of experiments;
- auto restarting the experiment or some part of experiment in case of failure;
- use of database for experiment creation and assigning automatic experiment identity;
- and the ability to reproduce the completed experiments fully or partially.

The current version of Autosubmit has an object oriented design and uses Python as its programming language. It can submit a sequence of jobs with different parameters to the queue of different queueing systems (such as PBS, SGE and SLURM). All the jobs have a common template, which is filled with different parameter values and submitted as jobs to the queue. Autosubmit acts as a wrapper around the scheduler, monitoring the number of jobs submitted or queuing and submits a new one as soon as a space in the queue would appear until the entire sequence of jobs is submitted. The most interesting capability is that Autosubmit can deal with the dependency between jobs. (i.e.: it can wait for a particular job to finish before launching the next one). Autosubmit can manage different types of job with different templates. Autosubmit can also restart a failed job, stop the submission process and restart where it left it. It stores a minimal amount of information into a SQLite database.

2.11. Dynamic memory allocation and load-balanced domain decomposition

NEMO uses a regular, two-dimensional domain decomposition of its regular latitude/longitude mesh. Computation is performed at all grid points and the results masked so that results at land points are set to zero. As a consequence, the code appears to be well load balanced but the computation on land points is redundant. Some of this redundancy can be removed by using a pre-processing tool to identify land-only processor domains and eliminate them.

STFC's Computational Science and Engineering Department has considerable previous experience with optimising the Proudman Oceanographic Laboratory Coastal Ocean Modelling System (POLCOMS). This code has a few significant differences from NEMO:

- Each sub-domain is assigned to one MPI process, not equally sized;
- Uses a multi-core aware recursive k-section partitioning algorithm for balancing work load (on basis of number of sea points);
- Uses 2-d wet/dry masks to avoid redundant computation on land points
- Model variables are 3d arrays with level index first (NEMO has level index last);
- Dry points are trimmed from halo exchanges, reducing the amount of data that must be sent in MPI messages.

We suspect that these differences are partially responsible for the anecdotal reports by scientists using both NEMO and POLCOMS of the latter's better performance. (There are likely to be differences in the scientific approach of the two codes that also contribute to the difference in performance.) The aim of this work therefore is to implement the above features within NEMO and assess their effect on the performance of the code, especially in model configurations containing a large number of land points. As this project is not part of PRACE-1IP and has not yet finished, there are not yet results to be reported.

3. Conclusions

The EC-EARTH earth system model consists at the moment of the IFS atmosphere model and the NEMO ocean and sea-ice model, coupled together with the OASIS coupler. These three components run as one MPMD program. Due to the separate development teams for all components, the EC-EARTH package is quite complicated to port to new architectures. This has been significantly simplified by the new compilation system in EC-EARTH 3.1, which unifies the different build mechanisms into one XML file. The EC-EARTH model is ported to different architectures, including the Intel-based Curie system and the IBM Power6-based Huygens system. Different configurations with a resolution up to T799 for IFS and 0.25 degree for NEMO (about 25km for both components) were tested on the Curie platform. Many parts of the IFS model are only activated at specific intervals and the time per timestep can therefore vary considerably. This is a complicating factor when looking for a balanced combination of IFS and NEMO tasks. The scalasca tool was used to separate the performance of each component. It is best to choose enough MPI tasks for the NEMO model, so it is fast enough to keep up with the shortest interval between two coupling timesteps in IFS. An analysis of the different timesteps in IFS shows that the computational timesteps (without radiation, coupling or I/O) are the bottlenecks for further scalability on Curie. The performance of the IFS model depends critically on the MPI_Alltoallv call, which is in general quite slow on Curie and does not scale beyond 1000 MPI tasks. A simple benchmark shows that the use of OpenMP in IFS could benefit the MPI communication overhead. Although IFS supports OpenMP, we did not succeed to use it effectively on Curie. Although the OASIS coupler is not the major bottleneck, it is expected to be when the scaling increases. The OASIS team changed the coupled from a stand-alone program to a library linked with the IFS and NEMO components, which diminishes the coupling overhead. A built-in FFT routine in IFS was replaced with a generic call to the FFTW3-interface, which enables the use of platform-optimized FFTW-routines, e.g. MKL or CUDA. The distribution of MPI processes across the nodes does not change the performance. A description of the I/O subsystem

in IFS is included as a first step to improve its I/O performance. The Darshan tool was used to get a good idea of the total I/O in EC-EARTH, both of the genuine output files and logs. Different verbosity options were described to decrease the amount of metadata operations due to log updates. The development of a framework for ensemble simulations and a more flexible domain decomposition are also described.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-211528 and FP7-261557. The work is achieved using the PRACE Research Infrastructure resources Curie (CCC, France), Huygens (SARA, The Netherlands) and Ekman (PDC, Sweden).

References

1. W. Hazeleger et al: EC-Earth V2: description and validation of a new seamless Earth system prediction model, Climate Dynamics, 2011, DOI: 10.1007/s00382-011-1228-5
2. G. Madec et al: NEMO ocean engine, Note du Pôle de modélisation de l'Institut Pierre-Simon Laplace No 27, ISSN No 1288-1619, January 2012.
3. S. Valcke, OASIS3 User Guide (prism_2-5). PRISM Report Series, No 2, 6th Ed., 2006.
4. The Scalasca Development Team, Scalasca 1.3 User guide, http://www.scalasca.org, March 2011.
5. http://www.mcs.anl.gov/research/projects/darshan/
6. J. Donners et al: Towards petascaling of the NEMO ocean model. Poster, EGU 2012, Vienna, April 23-27th, 2012, http://presentations.copernicus.org/EGU2012-12983_presentation.pdf.
7. Chandan Basu: Improving MPI communication latency of Euroben kernels, PRACE-1IP white paper, 2012.
8. Alastair McKinstry: possible use of GPGPUs to calculate FFTs, PRACE T7.5 white paper, 2012.
9. Eric Maisonnave: Ec-Earth – OASIS-MCT, presentation 7-12-2011, https://redmine.dkrz.de/collaboration/attachments/804/oasis_mct.pdf
10. Lecointre et al: Calcul massivement parallèle dans un contexte de modélisation océanique, Séminaire réseau calcul Grenoble, 28-03-2012
11. Wautelet and Kestener: Parallel IO performance and scalability study on the PRACE CURIE supercomputer, PRACE-1IP white paper, 2012. Available online: http://prace-ri.eu/IMG/pdf/Parallel_IO_performance_and_scalability_study_on_the_PRACE_CURIE_supercomputer.pdf
12. Dennis et al: An Application Level Parallel I/O Library for Earth System Models, Int. J. High Perf. Comput. Appl., 2011, accepted. Available online: http://nldr.library.ucar.edu/repository/assets/osgc/OSGC-000-000-003-873.pdf

Appendix A: awk script for IFS timestep analysis of EC-EARTH output

```
# This script can be saved in a file script.awk and used with the command:
#     awk -f script.awk < IFS-output.txt
   BEGIN {
   #These are parameters that need to be adjusted to your specific simulation
   .
     iofreq=30;
     cplfreq=15;
     radfreq=5;
     lasttimestep=60;

   #These parameters should not be adjusted
     compfreq=1;
     totalsteps=0;
     init("io",iofreq,"IO");
     init("cpl",cplfreq,"Coupling");
     init("comp",compfreq,"Computing");
     init("rad",radfreq,"Radiation");
   }

   function init(type,frequency,text)
   {
     time[type]=0;
     steps[type]=0;
     mintime[type]=9e99;
     maxtime[type]=0;
     name[type]=text;
     freq[type]=frequency;
   }

   function update(type)
   {
     time[type]+=$7;
     steps[type]+=1;
     totalsteps++;
     mintime[type]=mintime[type]>$7?$7:mintime[type];
     maxtime[type]=maxtime[type]<$7?$7:maxtime[type];
     next;
   }

   function printtype(type) {if (steps[type] != 0) {
     printf("%10s %s %3d %s %5.2f %s %5.2f %s %5.2f %s¥n",name[type],"steps: ",steps[type]," Time: ",
(time[type])/(steps[type]), " [", mintime[type],"..", maxtime[type], "]")
   } else {
     printf("%10s %s %3d¥n",name[type],"steps: ",steps[type])
   }}
```

```
  #Skip the first, last and next to last record, as these contain initialization and finalization
routines.

  / STEP  / && ($3==0 || $3==lasttimestep-1 || $3==lasttimestep) {next}
  / STEP  / && ($3%freq["io"]==0)  {update("io")}
  / STEP  / && ($3%freq["cpl"]==0)  {update("cpl")}
  / STEP  / && ($3%freq["rad"]==0)  {update("rad")}
  / STEP  / && ($3%freq["comp"]==0) {update("comp")}

  END {
    printtype("comp");
    printtype("cpl");
    printtype("rad");
    printtype("io");
    print "Total steps      : " totalsteps
  }
```

Appendix b: Changes to the routine ifs/control/cnt4.F for manual source code instrumentation of different IFS timesteps.

```
...
#include "epik_user.inc"
  EPIK_USER_REG(long_name,"iterio")
  EPIK_USER_REG(short_name,"itercomp")
  EPIK_USER_REG(mid_name,"itercpl" )
...
DO JSTEP=NSTAR2,ISTOP
IF (MOD(JSTEP-NSTAR2,6).eq.0 .and. JSTEP.ne.NSTAR2) THEN
  EPIK_USER_START(long_name)
ELSEIF (MOD(JSTEP-NSTAR2,6).eq.3) THEN
  EPIK_USER_START(mid_name)
ELSEIF (JSTEP.NE.NSTAR2) THEN
  EPIK_USER_START(short_name)
ENDIF
...
  IF (MOD(JSTEP-NSTAR2,6).eq.0 .and. JSTEP.ne.NSTAR2) THEN
    EPIK_USER_END(long_name)
  ELSEIF (MOD(JSTEP-NSTAR2,6).eq.3) THEN
    EPIK_USER_END(mid_name)
  ELSEIF (JSTEP.NE.NSTAR2) THEN
    EPIK_USER_END(short_name)
  ENDIF
ENDDO     ! End of main loop over time steps
...
```

Appendix c: Code to measure MPI_Alltoall performance

```
      program mpimania

! tries to stress network by calling mpi_alltoall
! repeatedly
!
      implicit none
      include 'mpif.h'

! totalsize=the total size of the data sent by all MPI ranks
      integer, parameter :: totalsize=675000000

! n=the size of the send (and receive) buffer at each MPI rank
! chunksize=size of one message
      integer chunksize, n
      integer,parameter :: times=100
      integer,parameter :: repeats=5
      integer msize,myid,ierr,i,time2,r
      integer, allocatable :: a(:),b(:)
      real*8 t0,t1
      character*(MPI_MAX_PROCESSOR_NAME) name
      integer lname

      call mpi_init(ierr)
      call mpi_comm_size(mpi_comm_world,msize,ierr)
      call mpi_comm_rank(mpi_comm_world,myid,ierr)

      call MPI_Get_processor_name(name,lname,ierr)

      print *,myid,": ",name(1:lname)

      n=totalsize/msize
      chunksize=n/msize
      allocate(a(n))
      allocate(b(n))

      if (myid .eq. 0) then
        print *,'arraylength =         ',n
        print *,'number of alltoall  = ',times
        print *,'number of processes = ',msize
        print *,'chunksize           = ',chunksize
      endif

      do i=1,n
        a(i)=100*(myid+1)+i
      end do
```

```
! first few calls to get rid of possible startup effects
! this is one more call than the
!   MCA btl: parameter "btl_openib_eager_rdma_threshold"

  do i=1,17
      call mpi_alltoall(a,chunksize,mpi_integer,b,
     &    chunksize,mpi_integer,
     &    mpi_comm_world,ierr)
      end do

      do r=1,repeats
      call mpi_barrier(mpi_comm_world,ierr)
      t0 = mpi_wtime()

      do i=1,times
        call mpi_alltoall(a,chunksize,mpi_integer,b,
     &    chunksize,mpi_integer,
     &    mpi_comm_world,ierr)
      end do

      call mpi_barrier(mpi_comm_world,ierr)

      if (myid .eq. 0) then
        t1 = mpi_wtime()
        print *,'time per alltoall is:',(t1-t0)/times, ' (repeat ',r,')'
      endif
      enddo ! repeats

      if (myid .eq. 0) then
        print *,myid,'calling mpi_finalize'
      endif

      call mpi_finalize(ierr)
    end
```