Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Keeping computational performance analysis simple: an evaluation of the NEMO BENCH test

Stella V. Paronuzzi Ticco[a], Mario C. Acosta[a], M. Castrillo[a], O.Tintó[a]

and K. Serradell.[a1]

[a]*Barcelona Supercomputing Center*

C/Jordi Girona 29, 08034-Barcelona, Spain

Abstract

In 2019 a non-intrusive instrumentation of the NEMO code, aimed to give information about the MPI communications cost and structure of the model, was created by E. Maisonnave and S. Masson at the LOCEAN Laboratory, in Paris. The main goal was to identify which developments have to be prioritized for the model to enhance its scalability: a new NEMO configuration, called BENCH, was specifically developed for the purpose, offering some easy way to make performance measurements, and hoping to simplify future benchmark activities related to computing efficiency.
In the first part of this work we study if this configuration can actually be used as a valid tool to get insight into NEMO's performance, and then proceed to use the BENCH test to study some of NEMO's most known bottlenecks: I/O and thtime e north fold. Additionally, we take the chance to investigate a topic that is gaining popularity among NEMO developers: the variability of time required to perform a time step and how it influences NEMO's performance, obtaining prescriptions on how to avoid or mitigate such behaviour.

Introduction

The Nucleus for European Modelling of the Ocean (NEMO) [4] is a state-of-the-art modelling framework for oceanographic research, operational oceanography seasonal forecast and climate studies, which is used by a large community: about 100 projects and 1000 registered users around the world. It is controlled and maintained by an European consortium, made up by CNRS and Mercator-Ocean from France, NERC and Met Office from the United Kingdom and CMCC and INGV from Italy [5]. The NEMO framework allows various configurations based on the core ocean model OPA: global (ORCA grid) or regional, with various horizontal and vertical resolutions. To this core two more modules can be added: SEA-ICE, to compute thermodynamics, brine inclusions and sub grid-scale thickness variations, and TOP, for the study of oceanic tracers transport and biogeochemical processes. NEMO is parallelized by means of the MPI library and a domain decomposition method is used. The basic idea is to split the domain into several smaller domains and solve the set of equations by addressing independent local problems. Each processor has its own local memory and computes the model equation over a subdomain (or *tile*) of the whole model domain.

Since the configuration of this model can vary widely, to carry out a reliable study on the computing speed of the NEMO model, a performance analysis study should comprehend different resolution/module sets. It would be desirable though, to keep such a study the simplest possible, to avoid the necessity to have a deep knowledge in the physics of the model itself. This is the reason why a specific configuration, called BENCH [1], was made available among the tests in the last NEMO release,

[1] Corresponding authors e-mail addresses: stella.paronuzzi@bsc.es, mario.acosta@bsc.es, miguel.castrillo@bsc.es, oriol.tinto@bsc.es, kim.serradell@bsc.es

hoping to simplify future benchmark activities related to computing performance. It is based on the existing GYRE configuration, a simple cuboidal grid which does not include bathymetry (nor continents), and which dimensions can be easily changed by name list parameters. With respect to this, BENCH also adds the East-West periodical conditions and the North Pole folding.

This was not the first work trying to address computational problems with the parallel implementations of earth system models. Some illustrative examples could be [2], where a low-resolution climate model is analysed and major code modifications are proposed in order to increase the performance on a specific machine, and [3], where the different components of the Community Climate System Model are stress tested to find scalability issues. Another methodology that aims to help scientists working with computational models to deal with the difficulties they face, when trying to understand their applications behaviour was presented in [10]. In this methodology specific tools developed by computer scientists like Extrae and Paraver [11] were used to identify which parts of models present a bad performance and understand the role played by inter-process communication.

The aim of the work by Maisonnave and Masson was not to replace all these previous studies, but to develop a test that helped computer scientists to analyze and profile a complex framework like NEMO without the need of entering too much in the mathematical and physical details of the model. Though a noble goal, it was not clear if the test would provide valuable information on the original code, raising questions as: "is it a good representation of the real NEMO workload", "are the bottlenecks of the BENCH equivalent to the ones present in NEMO", "can we actually rely on the numbers we get from the test to extrapolate informations on NEMO". These are all questions that need an answer to understand if BENCH is a valid tool to carry on performance analysis on the NEMO model.

The goal of the performance activities carried on in this work goes further than just this: we give here a list that summarizes what is being presented in the paper:

1. In NEMO, MPI communications are wrapped in a small number of high level routines co-located in a single FORTRAN file. This code structure strongly facilitates a quick and non intrusive instrumentation for a study dedicated to the MPI communication pattern. Taking advantage of this, BENCH includes a feature to modulate the amount of MPI communications. With very little code changes the sequence of the communications can be modified or the communication block even switched off on purpose to estimate the effect of such modification to the model performance. We extend this feature to the SEA-ICE module.

2. We assess whether the BENCH test can be considered *computationally similar* to a standard NEMO run, and in this case, whether it can be used to study its performance.

3. Global configurations of NEMO use ORCA tripolar grids, which construct a global orthogonal curvilinear ocean mesh with no singularity points inside the computational domain. We proceed to measure the cost of producing model outputs when running on ORCA12 a grid with 1/12 degree in horizontal resolution, evaluating the impact on performance of the IO server XIOS. This is a "large size" grid, both in terms of computing resources and file size, and exactly for this reason is of the greatest interest assessing the impact it has on code's computational behaviour, as it could be the standard resolution to run in exascale facilities in the near future.

4. Finally, we evaluate the weight of the north fold boundary condition computation on an overall run. These so-called North-folding exchanges are made necessary by the particular tri-polar structure of the ORCA global grid and implies some extra halo communications, in a code where communications are already suspected to be the bottleneck that limits the model scalability.

Before proceeding with the results we give here a small description of the grids configuration that will be used, and technical specifics of the Marenostrum IV machine (MN4), i.e. where the tests are run.

Name	Horizontal resolution (degrees)	N of vertical levels	Total points
Orca2	2	31	0.8M
Orca1	1	75	7.9M
Orca025	1/4	75	110.4M
Orca12	1/12	75	991.6M

Table A: Summary of different available ORCA grid resolutions. The included information tells us the approximate separation between grid points near the equatorial line, the number of vertical levels, and the number of grid points that a specific resolution has.

ORCA is a grid family for global models whose common characteristic is its tripolarity, aimed to avoid numerical singularities at the North Pole. The geographic South Pole is conserved and from 80 ◦ S to 20 ◦ N, the grids are like a regular Mercator-grid, while in the north hemisphere two poles are used. From 20◦ N to the north-poles, the circles of the Mercator-grid are progressively distorted into ellipses. In the third dimension, the earth is assumed to be spherical and the vertical dimension is considered to be perpendicular to the surface. The resolution of these grids (Tab. A) is not homogeneous, and the denomination of the different grids refer to their resolution at the latitude at its coarsest part, that is the equator: ORCA2, ORCA1, ORCA025, ORCA12, etc.

All the scaling tests and runs presented here are carried out on the MareNostrum IV machine (MN4). It consists of 48 racks with 3,456 nodes and each node has two Intel Xeon Platinum chips, each with 24 processors, amounting to a total of 165,888 processors and a main memory of 390 Terabytes. With respect to the disk storage, it has a capacity of 14 Petabytes and is connected to the Big Data infrastructures of BSC-CNS, which have a total capacity of 24.6 Petabytes. A high-speed Omnipath network connects all the components in the supercomputer to one another.

As a final remark, note that, unless otherwise specified, the performance is always reported as *Simulated Years per Day* (SYPD). Although there are different metrics that can be used, being many of them field-specific, in the case of climate models this is the most widely accepted, as indicates the number of simulation years that can be completed in one day of wall-clock time and is an important measure of the ability to perform research in a timely manner. A general consensus is that 5 years per day is the minimum acceptable useful simulation rate [6].

1. Extending BENCH features

One of the appealing aspects of the BENCH configuration is that it offers the possibility to modulate the amount of MPI communications so as to evaluate their impact on NEMO's scalability [1]. In fact, NEMO code structure is such that all the communication routines are placed in a few files. This facilitates the study of the communication pattern and of the influence the amount of communication has on code's performance. Basically, the NEMO communication library provides functionalities for horizontal halo exchange and global operations between all horizontal subdomains. Whichever exchange among different domains necessarily needs to be done through these generic routines. Intervening on these few points in the code allows to modulate the overall amount of communications: for example, they can be made every two time steps, so as to study the benefits that would come from implementing a double halo algorithm; they can be even completely switched off [1]. This option, however, is available just for the OCE component (which models the ocean dynamics and solves the primitive equations), so as a first goal we extend this feature at least to the SEA-ICE module. This resulted a bit tricky since this feature is not tracked on any development branch of the source code: retrieving the list of changes needed to have this working just with OCE already showed to be not so trivial.

The BENCH test, as it is in the trunk of the NEMO repository, could be deployed without particular problems on the MareNostrum4 (MN4) platform: the architecture file was the one commonly used for

the different NEMO installations on MN4, and no ICE (SEA-ICE) nor TOP modules are included in the beginning. After following the instructions we got from E. Maisonnave, the modules `mpp_lnk_generic` and `mpp_nfd_generic` were modified. The goal was to run the model without calling any of the Routine_`LNK` functions, which are general purpose interfaces that manage all the communication between the subdomains in the time step loop. Since the SEA-ICE communication pattern does not include collective operations, no more actions were needed to extend the no-communications option to this configuration.

To give an example of the impact the communications have on the scalability of the model we run two different experiments with the BENCH, one with and one without communication, on an ORCA025 grid, and compare the results,

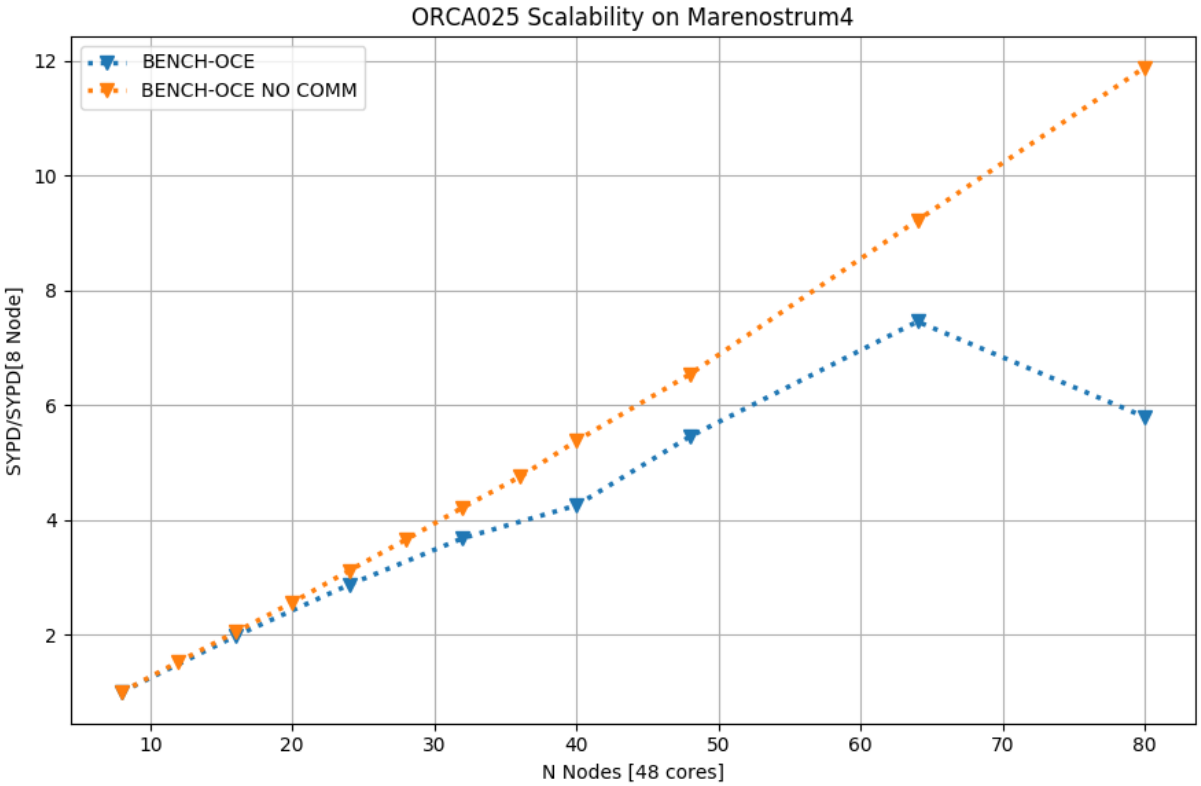

Figure 1: Scalability of BENCH with OCE component on an ORCA025 with, and without communication: at 65 nodes the difference in performance is around 25%, while at 80 nodes is already at 50%.

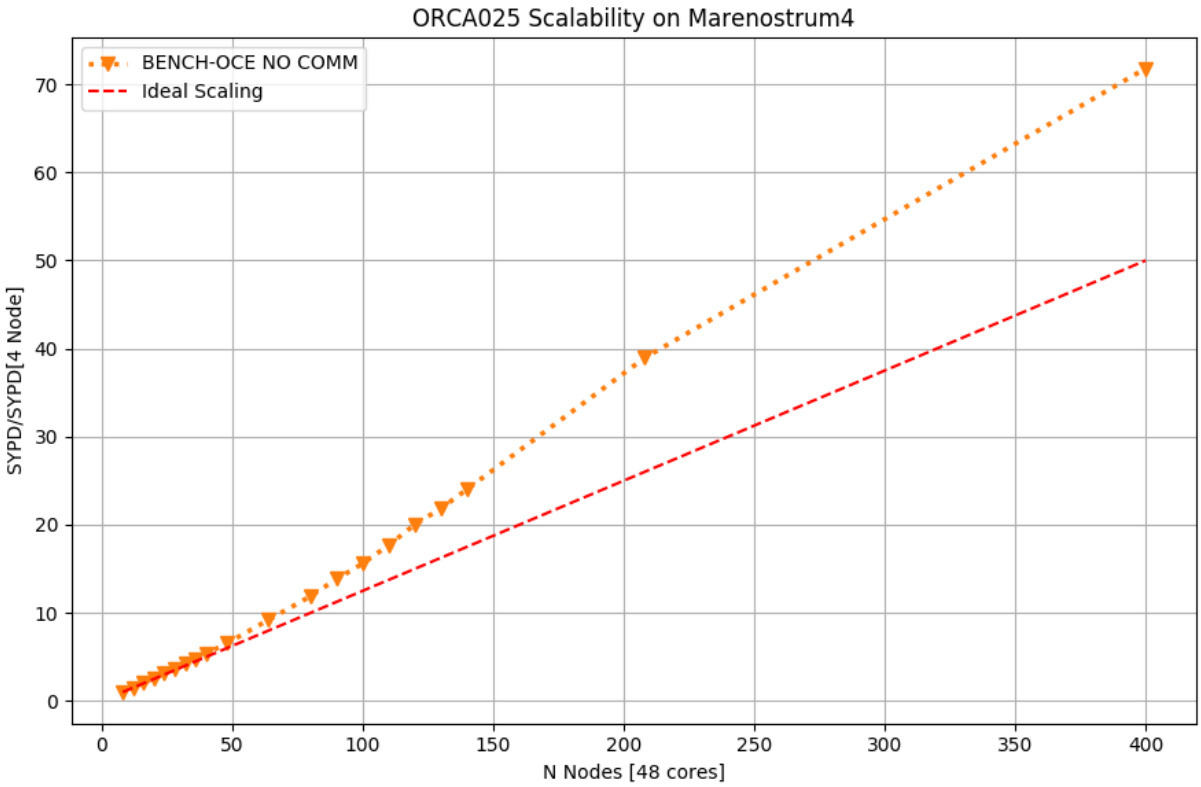

Figure 2: When communications are removed, the capabilities of the model to scale are fully exposed: the region of superscaling, which is present and starts around 50 nodes even in normal conditions, is here extended indefinitely.

Since when removing communication the model shows some numerical instability, following the advice of E. Maisonnave, we reduced the timestep of a factor 1/100. To be able to compare the throughput then, we rescale the results with their own value in the first point of the curve.
The obtained scaling curves (Fig.1) show how the communication overhead dominates and limits the scaling capabilities of the code. At 80 nodes we can measure a 50% increase in performance when communications are removed. Without this burden the code virtually never stops scaling: in Fig.2 the model, tested up to 400 nodes, shows no signs of degrading performance, and on the contrary it demonstrates that this would still be a region of superscaling.

2. Assessing similarity between NEMO and the BENCH test

The second and fundamental task was to study the feasibility of extrapolating conclusions on NEMO‘s computational performance starting from studies conducted on the BENCH test. With this aim we looked for a configuration which could be run in both cases: the optimal choice would be an ORCA1 grid (one of the smallest between the available grids in the ORCA family, but retaining the same number of levels as bigger configurations), so as to limit the number of parallel resources needed. This was impossible due to a lack of input data: this configuration could not be found anywhere for the NEMO4.0 version at the moment that this study was done. The second best choice was an ORCA2 configuration (2 degrees horizontal resolution), a little smaller than an ORCA1, but with less vertical levels (31 instead of 75). This configuration was not given for the BENCH, and we tried with little success to create one ourselves: after a few time steps we got numerical instability, thus making this config not viable either. We had to resort to an ORCA025 grid, for which we had both a name list input for the BENCH test, and the real input data for NEMO, with a considerable waste in time and computational power.

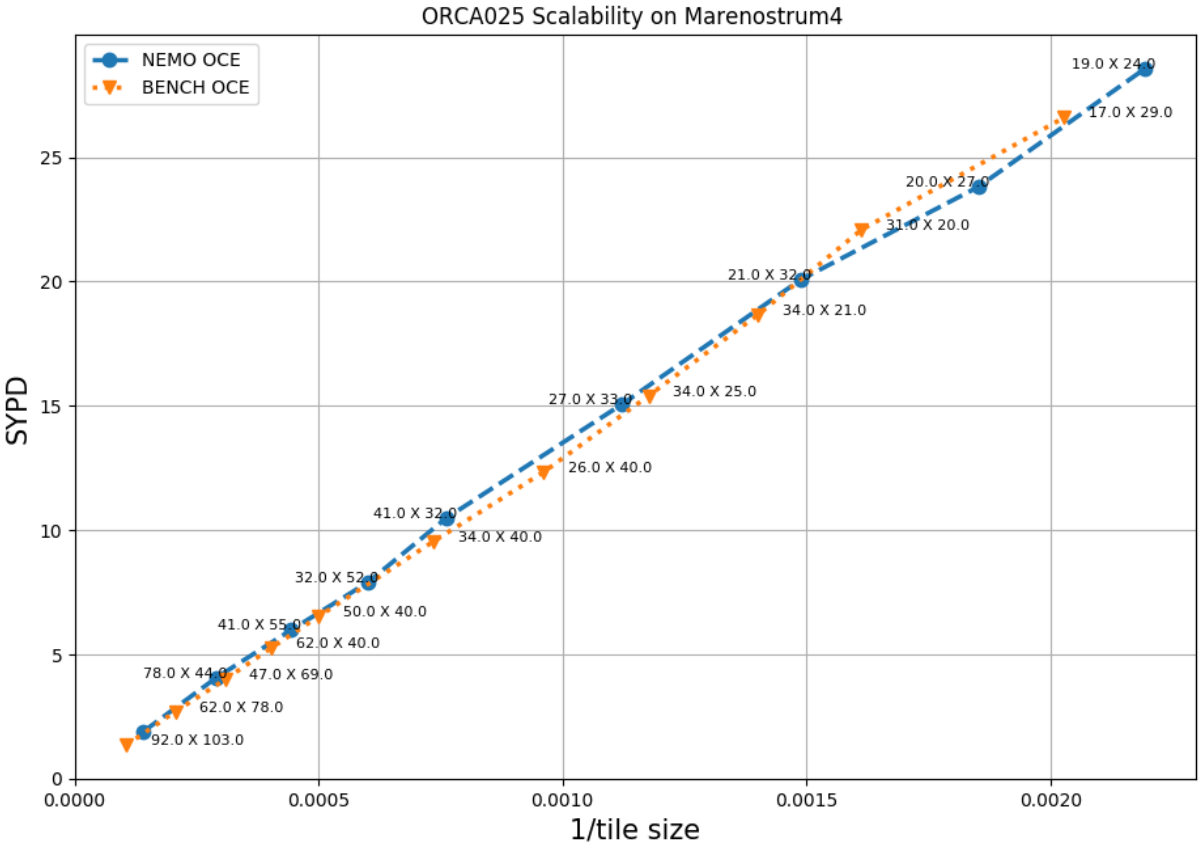

Figure 3: Scalability of NEMO using ~28 km grid resolution on MareNostrum4. Comparison of strong scaling of NEMO code with OCE component on an ORCA025 grid (75 vertical levels, 200 s. time step) with the BENCH test run on the same grid.

Number of Cores	NEMO Efficiency	BENCH Efficiency	Number of Nodes
384	1	1	8
576	0.97	0.98	12
768	0.97	0.97	16
960	1.02	0.96	20
1440	0.98	0.94	30
1920	0.98	0.91	40
2400	0.93	0.91	50
3360	0.84	0.93	70

Table B: The data used to deploy the scaling curve in Fig.1: the number of processors varies for a fixed total problem size, and in both the case of BENCH and in the case of NEMO the efficiency shows a similar behaviour.

To assess whether the BENCH test can be considered *computationally similar* to a standard NEMO run, and in this case, whether it can be used to study its performance, we generated the scaling plot we present in Fig. 3. Both configurations can be considered similar as long as we measure the throughput as SYPD with respect to grid points per core. The results are shown as SYPD per tile size, and not as SYPD per core or node, as it is commonly done. The reason is that the two configurations differ in a fundamental aspect: the BENCH is not performing the 'land removal process', i.e. the removal of sub-domains entirely constituted of land points, since it has no bathymetry nor continents. This, on the same number of cores, will result in a different domain decomposition: thus a different load per core, resulting in scaling curves with different slopes. This way of presenting the data allows for showing that the two models have the same scaling properties.

The results show that for both the case of BENCH and the case of NEMO the SYPD are similar, and how this new configuration reproduces quite accurately the computational property NEMO displays at the same resolution. We report on Tab. B some additional data that show how the efficiency is also similar in the two cases. These data highlight how BENCH can be used for the purpose it was designed for, and candidate this configuration as a very powerful tool to be used when small knowledge on the model is given but still one wants to have some insights on performance.

2.1 The time steps variability

At this stage of the work we took the chance to investigate a topic that is gaining popularity among NEMO developers. Many noted how optimizing the model does not always lead to better performance and were able to link this to the variability of the computational time needed to perform a time step, generically visualized as a kind of noise in their length [7].

In order to investigate the problem we studied the efficiency of the model when progressively reducing the number of cores per node: the reason is that NEMO is a memory-bound code, and from some benchmark we run, we knew that the band saturates long before the node is filled. For this reason, we expect that lowering the number of resources will not correspond to a loss in performance.

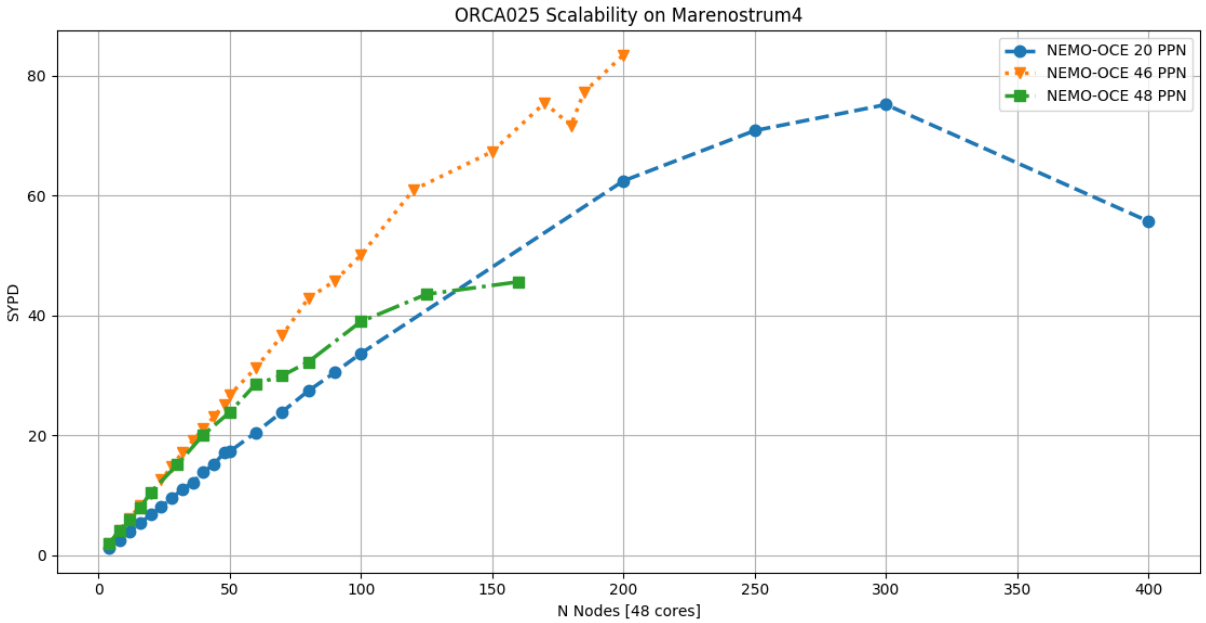

Figure 4: The three different scaling curves that are obtained with a different number of filled cores per node: filling the node completely (green squared line) bring to a bad scaling already around 150 nodes, while leaving two empty cores per node brings good scaling properties up to 200 nodes.

We took a number of scaling curves: starting from 20 cores per node and going upwards, we distributed resources equally among sockets (thus leaving part of the node empty). Then we measure the throughput and compare the results.

In Fig. 4 we can see that increasing the resources from 20 to 46 cores per node corresponds to an increase in performance, highlighting that NEMO is not totally memory-bound. While filling the node corresponds to a performance drop: we reach the best compromise between performance and machine usage, for the MareNostrum 4 facility, when 46 processes per node are used (orange dotted line), as this brings the best number of SYPD for the same amount of resources.

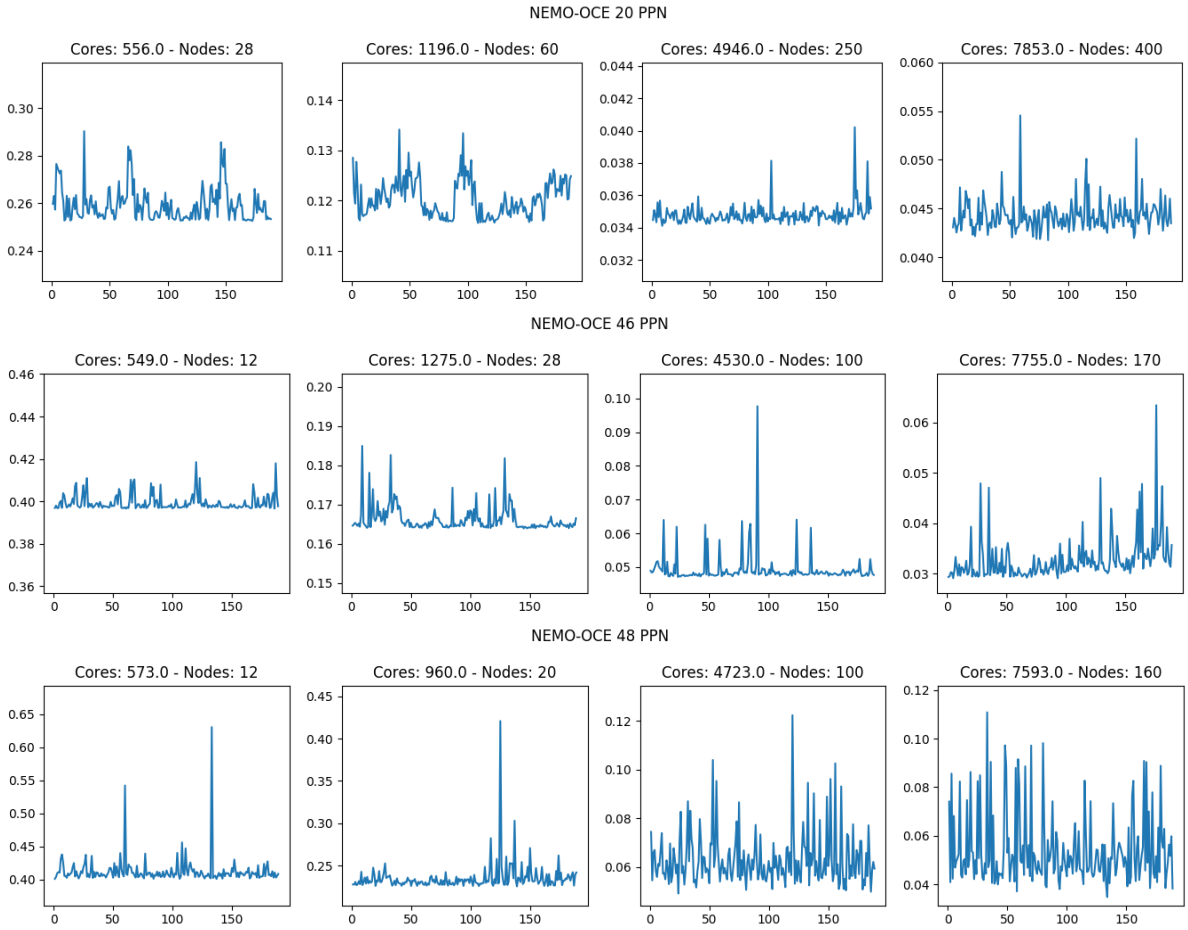

Figure 5: The walltime for executing a single time step, in seconds, when changing the layout of processes on the nodes, and increasing the number of nodes. From top to bottom: 20 cores per node, 46 cores per node, and fulfilling all the available resources (48 cores per node).

Considering these results, and that it was already suspected that NEMO has a problem with noise in time steps length at a high number of cores we proceed to check if these two facts can be related. Looking at the columns in Fig. 5 it can be noted how, as the number of cores grows, the variability in the time needed to perform a time step also increases. However, comparing the rows of the same figure, it can be ascertained that a fundamental role in this is played by the number of resources used per node: the amount of the processes on the sockets influences this phenomenon, and fulfilling all the available cores leads to a great variability. Looking at the first and second rows of Fig. 5 we can see that the time step length variability is not so big as in the third row; moreover, it does not increase too much as the number of cores increase. Since between 20 cores per node and 46 cores per node, we reach the point where the bandwidth saturates, and this does not lead to an increase in the variability, we can point the finger at the operating system instead as probable culprit for the noise.

To give some numbers: running an ORCA025 resolution on 160 nodes (7680 cores) leads to 45 SYPD using all the available resources, while to 67 SYPD when leaving one core empty per socket on each node. Leaving the half of every socket empty (blue dotted line in Fig.3, and first row of Fig.2) leads to a small variability that does not compensate for the waste in computational resources and the increase in communications overhead, eventually not being beneficial. Moreover, in the latter configuration we can detect good scaling performance up to 9.000 cores, while in the first case already at 6.000 cores the code does not scale anymore.

3. XIOS I/O server impact on performance

One of the goals set at the beginning of the project was to assess the cost of producing output when running at ORCA12 resolution. The output in this framework is handled by the XML Input/Output Server (XIOS), an asynchronous MPI parallel I/O server [8] [9] that is used by Earth system models to avoid contention in the I/O and is developed by the IPSL with an Open Source CEA CNRS INRIA Logiciel Libre (CeCILL) License. It focuses on offering high performance to achieve very high scalability with support for high-resolution output and for temporal and spatial post-processing operation.

This step took a considerable time, since we found that in a lot of configurations (i.e. #*N* nemo process + #*M* XIOS server, per node) the runs were failing with an MPI abort, due to some unmatched MPI call. This was happening both with XIOS2.0 and XIOS2.5, the two standard versions of XIOS used by the community at this moment. All the measurements and results shown so far were not taking into account the burden implied when the output process is enabled. We measured how costly it would be to run the BENCH configuration on an ORCA12 grid, without ICE.

NODE-BENCH-XIOS processes	No XIOS [s]	XIOS2.0 [s]	XIOS2.5 [s]
30 - 40 - 1	0.4	0.02	X
30 - 40 - 2	0.4	0.03	X
60 - 20 - 1	0.7	0.04	0.1

Table C: First column reports the number of nodes used, and for each node the number of BENCH and XIOS processes used. The results are displayed as seconds elapsed for time step. All the reported results are taken in "one file" mode, i.e. not using NetCDF parallel library, as this would bring a further slowdown of ~30%. The results marked with an 'X' are experiments run out of memory.

We increased the output frequency to one record per time step on purpose, so as to increase the impact on performance. We tested the two versions and in Tab. C reports the steps per second measured with the different configurations, fixing the number of BENCH processes to 1200 and changing the way the processes are distributed on the nodes.

In Tab. C we compare the seconds spent in an XIOS time step with the seconds spent in a normal one, i.e. the duration of a time step that performs output with one that does not: they are from six to twelve times slower. It is important to remark that all the reported results have been taken in "one file mode", meaning that we are not using the netCDF parallel library, but the serial version instead. This is because the "multi-file mode" was adding an extra overhead to the computation, requiring even more resources to be able to get to the end of a run. It is probably true that the frequency at which the data were taken is too high and we are measuring such a high downfall in performance because one output each time step is just too demanding for XIOS. Even, if in the frame of this project we did not have time, nor resources to investigate different configurations, it is probably worth exploring some different frequencies, and we plan to do so in the near future.

4. The north fold execution

The last planned assignment for this work was to evaluate the weight of the north fold boundary condition computation. Created to handle the north boundary of the ORCA tripolar grid, are made necessary by the particular tri-polar structure of the ORCA global grids: the uppermost subdomains located along the North Pole exchange their north-south halos between them. This particularity, combined with the structure of variables, adds extra array copies and extra communications to the halo

exchanges of the North fold area. Up to four lines, instead of one, are communicated: our goal here is to assess the impact such extra communication has on the computational performance.

For this task we took advantage of the feature present in BENCH to disable the communication.

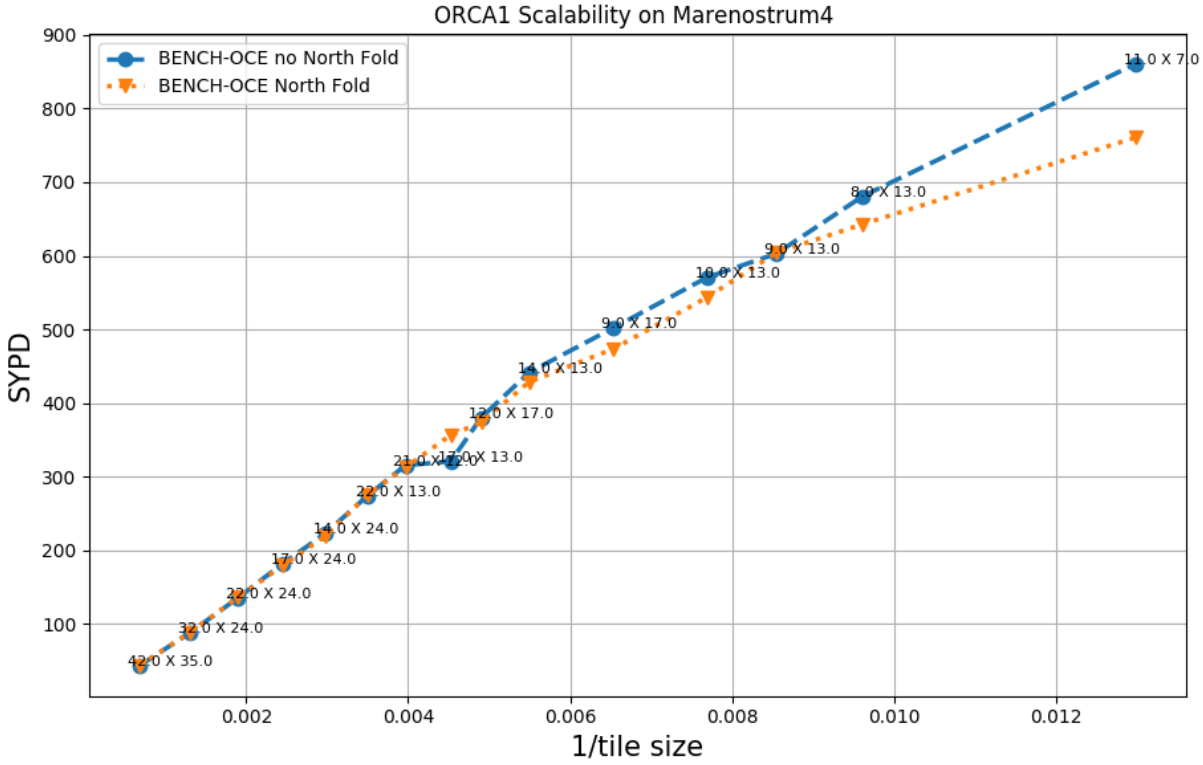

Figure 6: Scalability of BENCH using ORCA1 grid resolution on MareNostrum4: the blue dashed line represents the runs in which the North Fold communication have been disabled, while the orange dotted curve depicts the data we got with "normal" runs.

Inside the subroutine `ROUTINE_NFD` in the file mpp_nfd_generic.h90, which hosts the interfaces of the communication subroutines related to the north fold communications, we put a return statement. Then we proceed to compare the resulting scaling curve, for an ORCA1 grid, with a curve taken with normal runs. The results are shown in Fig.6: there is around 2% difference between the two curves measured at 25 nodes, rising to around 10% at 60 nodes.

Conclusions

We have shown how the BENCH test can accurately represent the computing performance of NEMO itself, giving a correct way of interpreting the data that may spare the experts working on optimization tasks to struggle with the search for input data or with domain decomposition issues. What we think is still lacking, and would have sped up our work significantly, is a working ORCA2 grid configuration for the BENCH.

In section 2.1 we also reported a study that illustrates how filling completely the nodes leads to a loss in performance due to a sort of noise in the time steps duration, and explained how this can be related to the need of leaving some space free for the operating system to work. Eventually we conclude that the best way to run NEMO, in all his flavours, on the Marenostrum IV HPC is to use 46 cores per node.

With respect to the I/O evaluation part, we found out that, when running on an ORCA12 configuration, the time steps involved in output operations will slow down with respect to a *computation-only* one, from six to twelve times. The XIOS version used is also crucial: it looks like the newest version needs around double the memory compared to the previous one. Not exploiting the NetCDF library parallel capabilities also seems like a good idea. Finally, it is probably worth exploring some different

configurations, changing the frequency at which the output is dumped on files, and for sure, we will do so in the near future. It is likely in fact that performing output at each time step is too demanding for XIOS, which is not performing well because it was not designed to work under such high stresses.

Finally, as what concerns the north fold extra communications, it seems like they become a burden for the model at a high number of cores, a region where the communications in general are already the bottleneck. In this sense, it is a further remark of the fact that whichever attempt to optimize the model should necessarily target this topic.

References

[1] Nemo 4.0 performance: how to identify and reduce unnecessary communication, E. Maisonnave, S. Masson

[2] P. Hanappe, A. Beurivé, F. Laguzet, L. Steels, N. Bellouin, O. Boucher, Y.H., T. Aina, M. Allen, FAMOUS, faster: using parallel computing techniques to accelerate the FAMOUS/HadCM3 climate model with a focus on the radiative transfer algorithm, Geosci. Model Dev. Discuss. 4 (2011) 1273–1303.

[3] R. J Haarsma, M. J Roberts, P. Vidale, A. Catherine, A. Bellucci, Q. Bao, P. Chang, S. Corti, N. S. Fukkar, V. Guemas, J. Von Hardenberg, W Hazeleger, C. Kodama, T. Koenigk, L. Ruby Leung, J. Lu, J.J. Luo, Jiafu, Mao, M: S. Mizielinski, R. Mizuta, P. Nobre, M. Satoh, E. Scoccimarro, T. Semmler, J. Small, and J: Von Storch. High Resolution Model Intercomparison project (HighResMIP v1.0) for CMIP6. Geoscientific Model Development, 9(11):4185–4208, 2016.

[4] G. Madec, NEMO Ocean Engine, No. (27), 2015.

[5] www.nemo-ocean.eu.

[6] J. M. Dennis, M. Vertenstein, and A. P. Craig. Performance Evaluation of Ultra-High-Resolution Climate Simulations. In 10th LCI International Conference on High-Performance Clustered Computing, 2009.

[7] Oral communication during NEMO Developer's Comittee

[8] E. Maisonnave, I. Fast, T. Jahns, J. Biercamp, Y. Meurdesoif, and U. Fladrich. CDI-pio & XIOS I/O servers compatibility with HR climate models. Tech. rep. 2017, p. 19.

[9] M. Hanke, J. Biercamp, C. Escamilla, T. Jahns, D. Kleberg, P. Selwood, and S. Mullerworth. Deliverable 7.3 – Reference implementations of Parallel I/O and of I/O Server

[10] O. Tintó Prims, M. Castrillo, M. C. Acosta, O. Mula-Valls, A. Lorente, K. Serradell, A. Cortés, F.J. Doblas-Reyes, Finding, analysing and solving MPI communication bottlenecks in Earth System models, Journal of computational Science, Vol 38, 2019

[11] Tools@bsc.es. EXTRAE User Guide.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EU's Horizon 2020 Research and Innovation programme (2014-2020) under grant agreement 823767.