IS-ENES3 Deliverable D4.4

Report on NEMO Quality Assurance Update

Reporting period: 01/01/2022 – 31/03/2023

Authors: Claire Lévy (CNRS-IPSL), Italo Epicoco (CMCC), Mario Acosta (BSC)

Reviewer: Andrew Coward (NOC)

Release date: 31/03/2023

ABSTRACT

The document reports the update of the NEMO Quality assurance process after the migration of the code from the SVN IPSL to the Gitlab Mercator repository. A key factor for the code quality assurance is also represented by its computational performance that should be evaluated after the code modification on a pool of target architectures. Two complementary approaches for the NEMO code computational performance analysis are here proposed and described in details.

Dissemination Level		
PU	Public	X
CO	Confidential, only for the partners of the IS-ENES3 project	

Revision table			
Version	**Date**	**Name**	**Comments**
1.0	23/01/2023	Italo Epicoco	First version sent to internal reviewer
2.0	28/03/2023	Italo Epicoco	Final version after internal review

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824084

Table of contents

1 Executive Summary

The continuous integration within the NEMO European ocean platform of software developments originating from several European institutes requires a rigorous and extensive quality control from the scientific and computational performance point of view.
Task 1 of WP4/NA3 contributed to improve unit and regression test frameworks (Trusting tools) and promoted their uptake and effectiveness in the community through improved modularity, documentation and installation procedures. Working with the community and understanding their needs across varied computing platforms, WP4/NA3 made continuous quality control available to the full development community leaving a legacy of sustainable, community-driven development. The scientific quality control of NEMO outputs is based on idealised test cases. Within this task, new test cases have been used to validate and demonstrate the benefit of existing and new features.
Moreover, Task 1 allowed us to design two complementary solutions that developers can adopt to test the computational performance of new versions of the NEMO code. The portability of the proposed integrated solution allows users/developers to have a report on the computational behaviour of the analysed code by running a NEMO test case, while a more detailed analysis can be performed by using the external profiling libraries considered in the second solution.

2 Objectives

This report covers the work done during the IS-ENES3 project to improve the quality control both in terms of tools and methodology.

Quality Control – Development Infrastructure
NEMO has been developed using Software Version Control (svn) on forge.ipsl.jussieu.fr up to November 2021. The move to Gitlab (forge.nemo-ocean.eu) hosted at Mercator OCean International has been a major effort for developers, associated with some obvious benefits allowing a better visibility of development efforts, so as the use of new efficient tools to interact with users, between developers and to move towards continuous development and integration processes. The choices of tools and their adoption is a community effort from NEMO developers. However, the technical installation and configuration of all the new tools has been done by the NEMO Systems Team and this work has been funded by IS-ENES3.

Quality Control – Monitoring Computational Performance
The NEMO code is used to study the Ocean evolution in several European project by thousands of users. The code is run on different platforms and computational performance are a key factor for the entire community. CMCC and BSC proposed two different tools to execute a performance analysis of new developments. The design work has been funded by the IS-ENES3 project, it has been completed in this Task and reported in the present deliverable, while the tools will be finalized and integrated in the NEMO framework as unfunded activity performed by the NEMO partners by the end of 2023.
Sections 3, 4, 5, 6 and 7 reports about the scientific and technical quality control functionality integrated in the NEMO framework, while section 8 describes two approaches designed by CMCC and BSC for the code performance analysis.

3 Improving exchanges between users & developers: Discourse

The svn forum tools for information exchange were not user friendly. Following the move to Gitlab, we also decided to move to more recent and user-friendly tools. The NEMO users forums have been move to Discourse, allowing users to post their questions or information much more easily, and to get faster answer. In November 2022, these forums count 173 registered users (but are visible without any registration). Figure 1 reports a few indicators.

Figure 1 – Indicators about users interaction

(Note: DAU/MAU is the ratio of daily active users over the monthly active users)

4 Facilitating collaborative work between developers: Zulip

The NEMO developers' team is geographically distributed over all Europe. Regular videoconference meetings are organised every three weeks, as well as a one week meeting in person once a year. Still, there was a need to be able to exchange information, view what the others are doing, ask questions and get fast answers, etc… The Zulip tool has been set up to answer these needs.
The NEMO Zulip chat needs individual registration and allows sending information and questions, getting answers. It also includes automatic sending of information on all actions on NEMO Gitlab in the "Forges" stream. A few numbers in Figure 2 illustrate Zulip usage for NEMO development.

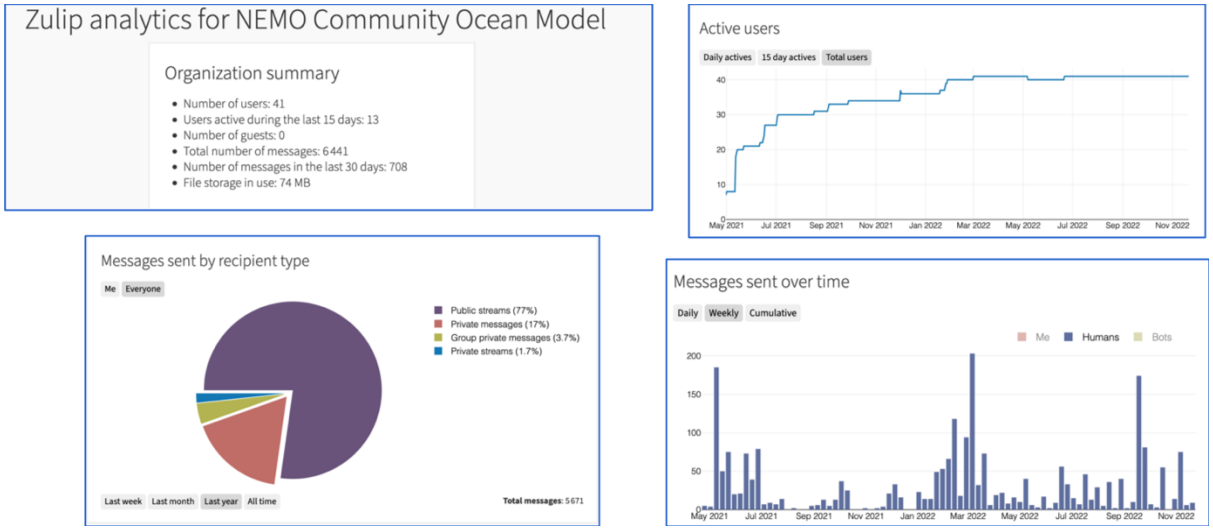

Figure 2 – Zulip usage for NEMO development

5 Improving the tool used for continuous testing: SETTE

Several improvements and new functionalities have been implemented in the NEMO integration testing tool SETTE. SETTE runs some short tests on each of the "Reference configurations" covering most of the NEMO features (from global ocean to 1D vertical column), allowing to ensure development verification (Are we building the development right?). The validation (Are we building the right development?) is expected to be demonstrated by the developer showing specific results of the development.

The complete history of SETTE recent developments can be found here https://forge.nemo-ocean.eu/nemo/nemo/-/commits/main/sette.
Brief summary of main features:

- Modified SETTE scripts to use the branch name as the default sub-directory for SETTE records (instead of MAIN). Also corrected a few minor issues and added a -u option to sette.sh, sette_rpt.sh and sette_eval.sh which avoids asking for user input (at least the looped input that is problematic in Continuous Integration applications)
- Add up to date arch files for recent target HPC computers (6 new platforms at NOC, ECMWF, Mercator…)

- Add support for running SETTE on “detached head” . This occurs when older commits are checked out for testing. A “detached head” is not associated with a branch (yet). These changes will, at least, recover an appropriate date for the commit being tested by SETTE rather than referring to the current origin of the branch (e.g. it will give a more precise history of the modifications being tested).
- Update SETTE scripts to find input files now available from the sette_inputs site
- Resolve "add a new "debug" (-d) option to `makenemo` and a specific Flexible Configuration Management FCM flag to switch more easily to debug mode"
- Improve the robustness of -n new configuration name) /- r (Reference configuration)/- a (Academic test case- options in makenemo.

5.1 Adding new “Reference configurations”

In relation with recent developments, the SETTE script has been updated to ensure that the new functionalities, or the forthcoming ones will be systematically tested:

- WED025 added: Weddell Sea, Antarctica in the perspective of upcoming developments on ocean-ice-atmosphere couplings
- Add SETTE '-a' option to activate Atmospheric Boundary Layer with ORCA2_ICE_PISCES (global ocean/ice/biogeochemistry at 2° resolution) "
- Add key_RK3 (coming new temporal scheme) now compatible with either key_qco (quasi Eulerian vertical coordinates) or key_linssh (fixed in time vertical coordinates)

6 Facilitating update of documentation: using Continuous Integration for User’s guide

The NEMO User’s guide[1] is the documentation on how to start with NEMO. It described, step by step how to download the code and to start using it whereas the NEMO Reference manuals[2] available on Zenodo describe the content of the code and how it is implemented.
As the first contact to NEMO for new users, the User’s guide needs to be fully up to date with NEMO release so as to make as easy as possible the first steps for newcomers.
This User’s guide covers these needs through *.rst files (ReStructuredText for the text content), namelist files and a gallery for the pictures. Using Sphinx, all these files are gathered to build the User’s guide.
They are implemented in a separate Gitlab project (and subprojects) so that as soon one or more files are changed, it triggers the Gitlab Continuous Integration process, generating both the pdf and the html version of the NEMO User’s Guide. However, the necessary separation of user-guide source material and NEMO code into separate gitlab groups means that updates to NEMO code can be made without enforcement of corresponding updates to the user-guide. It is left to the development protocols and a rigorous review to ensure that the two are kept in step.

[1] https://sites.nemo-ocean.io/user-guide/
[2] https://forge.nemo-ocean.eu/nemo/nemo/-/tree/4.2.0#id14

7 Moving towards continuous integration for NEMO codebase

The objective is to introduce a development protocol which define the procedure to regularly merge the code into a central repository. A key component of this procedure is to set up systematic and automatic testing of NEMO during a development and during the elaboration of a new release, using the SETTE script (see above Section 5).
A development branch[3] has been set up for this goal. Aside from making SETTE run in an automatic Continuous Integration (CI) process triggered at each new push for each development branch so as sending the appropriate return codes, it needs to convince each computing centre to accept job submission from an automatic process coming from forge.nemo-ocean.eu, which can still be a problem from some users.
As for now, the CI is about to be working on the development branch on the computing centre accepting it (not all of them yet). Our planning is to set it up systematically on future development branches, allowing a much more robust Quality Assurance in the future.

8 NEMO portable performance metrics

NEMO is constantly evolving; developers regularly work on improving different aspects of the model, adding new features and configurations, adapting the code for higher resolutions, etc.
It is possible that, during the development, some of the implemented changes create an unintended bottleneck in the model performance. Even though developers nowadays usually work along with HPC experts, discussing each modification regularly with them would not be feasible. It would result in a significant slowdown of the development process on both sides. On the other hand, making a single performance evaluation before a new version is released, it's not good either because it would be harder to identify the problem and reintroduce the changes correctly.
The best solution to this problem would be to allow developers to create baseline metrics. Later compare these metrics with those obtained for the development version, aiming to identify changes that affected performance.
Moreover, it is also important to allow the final users to have an idea of the code performance when some changes/improvements are introduced.
For all these reasons, it's necessary to implement an easier way for developers/users to profile NEMO. Two solutions are proposed by BSC and CMCC to solve the previously mentioned issues and are detailed in the following.

8.1 Wrapper based approach

The solution proposed by BSC is based on the development of a wrapper. A wrapper is a program whose principal purpose is to interconnect other programs or functions. In this case, the wrapper will connect NEMO and the required performance tools to generate a selection of fundamental performance analyses.
This wrapper design aims at being:

- Multi-platform: it must work on different HPCs with minimal parameter changes.
- Understandable: requires little knowledge of HPC to be used.
- Adaptable: be relatively easy to modify to fit every ESM.

[3] https://forge.nemo-ocean.eu/nemo/nemo/-/tree/61-continuous-integration-tests-with-sette

8.1.1 Methodology

In this section, we will explain the requirements and the general workflow designed to obtain performance metrics from NEMO and other ESMs. Figure 3 contains the graphical representation of the workflow.

8.1.1.1 BSC-Tools

To conduct performance analysis, we use an open-source set of tools developed at BSC. Therefore, those are an essential part of the wrapper's workflow.
The wrapper requires the following tools to be downloaded and installed from https://tools.bsc.es/downloads:

- Extrae
- Paraver
- Dimemas 5.4.2-devel
- Basic analysis.

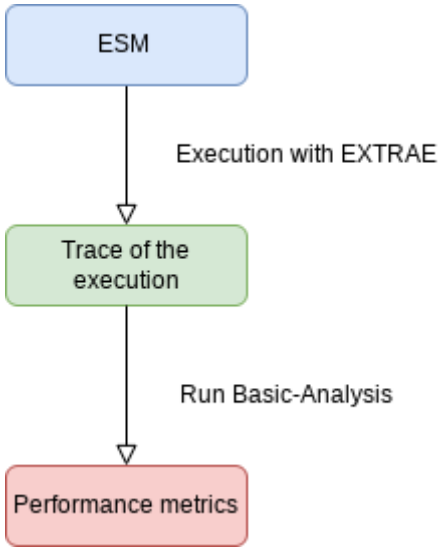

Figure 3 - Basic workflow of the elements connected to execute the performance analysis

Although the wrapper requires the installation of these tools, this should not be a problem for most of the HPC and modelling centres since BSC Tools are open sources and already installed as basic profiling tools is most of them. As an example, BSC Tools are available on Marenostrum, LUMI and LEONARDO supercomputers .

8.1.1.2 Workflow

First, we use Extrae, a tool that collects performance information on parallel applications at run time. After executing the NEMO, Extrae generates a trace file containing all the data collected during the execution.
A tool called Basic Analysis processes the raw data stored in the traces produced by Extrae and turns them into human-readable information. It also calculates performance metrics using this

data and those obtained from an execution simulation on an ideal machine. Finally, it stores the data in different formats suitable for the user.
The files generated are the following:

- Overview file "overview.txt": Contains a summary of the Basic Analysis tool execution. At the bottom, it shows the most relevant metrics, such as computational speedup, Instruction Per Cycle (IPC), CPU frequency, and global efficiency.
- Excel files (csv)
    - efficiency_table.csv: Contains all the metrics related to performance efficiency and generates scalability metrics only if at least two traces with a different number of cores are present.
    - modelfactors.csv: Contains the same data as "efficiency_table.csv", but the decimal values have six digits of precision instead of two.
    - other_metrics.csv: Contains other valuable metrics such as the speedup, number of processes, etc.
    - rawdata.csv: Contains all the data used to obtain the metrics.

- Plots (png)
    - efficiency_table-matplot.png: represents the metrics stored in efficiency_table.csv.
    - (2 traces minimum) efficiency-matplot.png: represents the evolution of efficiency with the increase of cores compared with the ideal efficiency.
    - (2 traces minimum) modelfactors-comm-matplot.png: represents the evolution of communication, transfer, and serialisation efficiency when increasing the number of cores.
    - (2 traces minimum) modelfactors-matplot.png: represents the evolution of all the efficiency-related metrics with the core increase.
    - (2 traces minimum) modelfactors-scale-matplot.png: represents the evolution of the scalability metrics when increasing the number of cores.
    - (2 traces minimum) speedup-matplot.png: represents the evolution of the speedup compared with the ideal speedup when increasing the number of cores.

8.1.2 Use case

This section describes the steps a user must follow to execute the wrapper designed for the NEMO basic profiling.
The project contains two main files, perf_metrics.sh (the executable) and perf_metrics.config (the configuration file). It also contains a Readme file indicating the requirements for executing the script.
Inside perf_metrics.config parameters for the wrapper execution are defined. Most have default values, but there are exceptions:

- Nemo_input_data: path where NEMO input data are located
- Nemo_path: NEMO installation path on your machine.
- Compilation_arch: name of the architecture file used in your environment for compiling NEMO.
- Modules: list of modules you need to load. If you want to load them manually, leave this parameter blank.
- Jobs_scheduler: scheduler installed in your machine. The currently supported queue managers are slurm, lsf, and torque.

Once the parameter file is set to your needs, you just need to execute the script using "./perf_metrics.sh ".
Once the script execution finishes correctly, it will store the metrics inside the Output directory, located outside the project folder by default. We recommend visualising tables and plots for a quick comparison of versions and using the excel files to compute the difference between versions in more detail.
In Table 1, you can see a typical output generated by the script, plotting different efficiency statistics for each number of cores. This table was created running the reference ORCA2_ICE_PISCES configuration removing the output.

Table 1 - efficiency_table-matplot.png generated by the wrapper using ORCA2_ICE_PISCES reference configuration, NEMO output files are disabled.

8.2 Integrated approach

The approach proposed by CMCC is designed to be easily integrated within the NEMO model and to allow both users and developers to compare the computational performance of different versions of the NEMO code. When integrated in the NEMO framework, the tool will automatically test the computational behaviour of a specific version of the model at different resolutions on any architecture where the NEMO code can be executed. It will produce as output a csv file that can be easily post-processed to extract needed information. Developers/users, already used to perform their tests with the SETTE package, will follow the same strategy to measure the computational performance of the code.

8.2.1 BENCH test case

The starting point for the design of the solution has been the BENCH [1] configuration, already part of the test cases available within the NEMO package, developed to measure the model performance. It allows us to (i) easily simulate the behaviour of the ORCA family configurations and (ii) activate/deactivate the different components such as biogeochemistry and ice, with or without using/producing input/output files, depending on the need to test the XIOS performance. Also the periodicity can be tested, with the advantage to analyse the model performance when the North Pole folding is activated. Finally, land-only subdomains can be

excluded when real configurations are executed with the BENCH test, simply providing the input files. For all these reasons, it will be used as the reference test case for the development of the performance automatic tool.
Anyway, the BENCH configuration cannot be used as is by the NEMO developers/users and it needs some adaptations.
In details, the BENCH configuration now uses the following files:

- the sh_bench, a bash script that takes as input the resolution of the model (chosen among the available ORCA family like 1°, 1/4° e 1/12°), the minimum and the maximum number of cores for the model execution. Intermediate runs depend on the number of cores per node on the target machine. The model can be launched on one of the machines listed and configured in the script, so that the script has to be updated whenever a new machine is added.
- the submit_bench, a second bash script called by the first one for each simulation. It creates and configures the execution directory for each run and submits the model. The submit_bench file creates a specific run script for the target machine depending on the job scheduler. The submit_bench script has to be updated if a new machine has to be added too
- the best_jpni_jpnj_eorca_<resolution>, three files (one for each resolution) containing the optimal parallel domain decompositions when changing the number of cores for model execution. The criteria to select the optimal decomposition is the minimisation of the domain size, in terms of number of grid points
- the namelist_cfg_orca<resolution>_like, three namelists, one for each model resolution
- the namelist_ref that allows us, among the other things, to activate/deactivate the HPC optimisations (i.e. tiling, extra-halo, neighbouring and North folding collective communications)
- the namelists for the available other components (i.e. ICE and PISCES)
- the cpp_BENCH.fcm file containing the keys to activate additional components (ICE and Biogeochemistry), XIOS and the loop fusion HPC optimisation.

Some of these files are ready to be used on different machines, while some others need to be updated by the final user, impacting on both the portability and the usability of the test case. Moreover, the current NEMO BENCH configuration cannot be launched through the sette.sh script and some changes are needed in order to fully integrate the tool into the SETTE package. The designed solution will replace both the sh_bench and the submit_bench by a new script called sette_test-perf.sh built starting from the sette_reference-configurations.sh, so very familiar to the final user. The batch-template files, already included in the SETTE package, will be used to automatically create the run_script for the target machine. The sette_test-perf.sh, executed by the sette.sh when the BENCH configuration is specified by the user, will run the model with the three resolutions (1, 1/4°, 1/12°). For each resolution, a set of experiments will be executed by increasing the number of cores starting from a minimum value to a maximum one. The step increase is done by the number of cores per node on the target architecture and the three parameters (minimum and maximum number of cores and cores/node) will be defined by the user within the sette_test-perf.sh. Each experiment outputs will be stored in the NEMO_VALIDATION dir of the experiment itself. However, a csv file (perf_report_csv) containing a set of performance metrics will be written by the sette_test-perf.sh script after the model execution and could be easily ingested in a database that can be used to track the performance changes with different versions of NEMO and architectures.

8.2.2 Metrics

Another aspect to take into consideration is related to the current performance metrics provided by the NEMO code after its execution. Two files reporting information about the model performance are provided: the timing.output and the communication_report.txt files. The timing.output file includes the following info:

- the total time for each simulation time step. The analysis of this metric allows us to estimate the impact of the I/O operations that are usually performed at predefined time steps
- the average time (computed on all the parallel processes) needed for executing the routines where the timing function is hard-coded activated
- the time spent by the root process (usually the slowest) for executing each routine
- the aggregate time spent for computation and communications by each process. Communications performed by the lbc_lnk routine, that implements the halo update, are separately considered by the collective ones, except for the neighbouring collectives used to perform the halo exchange. No synchronization among the parallel processes is implemented when the time spent in communication is monitored, so that some load unbalance effect could wrongly impact on the communication time. However, synchronization should be introduced to correctly evaluate time spent in communication and computation.

The communication_report.txt file reports about the number of communications executed during the run, about their nature (p2p or collectives) and the calling routine. This information is really useful for developers during the optimization activity, while information reported in the timing.output file gives an idea of the model behaviour on a target machine.
Some of these metrics will be maintained in the integrated tool, others will be synthesized and/or aggregated. Others will be added, such as:

- the I/O time that especially impacts on some time steps (depending on the I/O activation and the writing frequency)
- the memory footprint, monitored without instrumenting the model but using libraries, system calls or shell command lines, depending on the level of invasiveness we prefer to implement; it will be monitored when the run is starting, in the middle of simulation (avoiding IO time step), in the middle of simulation (considering IO time step), at the end of the simulation
- the energy consumption, that requires to be treated separately: the hardware performance counters can be used to evaluate the energy usage of the job. Some feasible solutions have been performed in task 3 of the IS-ENES3 WP4 project.

The final output will be a csv file that can be easily analysed and integrated in different contexts. The output file reports information about:

- the HPC system where the model has been executed (usually the compiler prefix used to identify the arch_file in the SETTE package)
- the NEMO version (i.e. the gitlab revision number)
- the model resolution (i.e. ORCA1, ORCA025, ORCA12)
- the number of parallel processes (number of cores used to execute the experiment)
- the date of the experiment execution
- the activated components as list of the comoilation keys
- the time spent to execute the fastest time step, called the minimum time step
- the time spent to execute the lowest time step, called the maximum time step
- the average time on all the time steps

- the list of the average times spent in the monitored routine
- the average I/O time computed as aggregation of the time spent in IOM routines from the previous list
- the list of routines calling p2p communications and the number of calls for each routine (extracted from the communication_report.txt file
- the list of routines calling collective communications and the number of calls for each routine
- the three values for the memory footprint (init, medium, final)
- the energy consumption (in joules) stored in the hardware performance counters and extracted through RAPL on Intel architectures.

The main advantage of the proposed solution is the fully integration in the SETTE package, already well known to the NEMO developers/users. Also the management of configuration and script files does not require any particular training and can be easily done to test performance of a NEMO version or to compare two different ones.

9 Conclusions and recommendations

NEMO developers pay particular attention to the updates and improvements of quality insurance. These efforts take some time. The main objective here was to improve a both reliable and simple development workflow. This goal can be considered achieved. It will allow the existing developers for faster and more reliable work. It also opens the way for a more visible and better organised offer for the users willing to contribute to NEMO development.

The approaches for monitoring performance proposed in this report, will be integrated in the NEMO framework in order to allow development teams to identify performance bottlenecks without requiring big effort. Moreover, they are designed to be easily incorporated into the development process, for example, when requesting a branch merge or releasing a new version. In this case, developers would only have to compare the metrics and verify that the performance did not decrease before applying changes.
The wrapper-based approach successfully achieved the main goals laid out in the introduction.

- The configuration file allows the users to adapt the wrapper for different platforms.
- Developers can easily use this tool because it only requires knowledge NEMO's compilation
- It's possible to modify the proof of concept to work on different ESMs because the core workflow does not depend on NEMO.

On the other hand, the integrated approach for measuring and monitoring the performance represents an extension of the SETTE package following the standard test procedure well known to NEMO developers and users. The portability of the solution is preserved by the integration of the test script within the structure of the SETTE package. Moreover, no installation of external tools is needed to execute the proposed performance tools.

10 Bibliography

[1] Improving ocean modeling software NEMO 4.0 benchmarking and communication efficiency (Irrmann, G., Masson, S., Maisonnave, É., Guibert, D. and Raffin, E.), In Geoscientific Model Development, volume 15, 2022.