IS-ENES3 Report

Nemo portable performance metrics

Authors: Carlos Peña de Pedro, Stella Paronuzzi, Mario Acosta.

Release date: *04*/11/2022

ABSTRACT

This deliverable proposes a methodology that eases Earth System Models (ESM) development teams on basic profiling. The objective is to detect unintended effects on performance

The report explains the workflow and tools used to perform the profiling, an overview of the data created, and a use case for guiding new users in executing the methodology.

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

Table of contents

Executive Summary

There is a need in Earth System Model (ESM) development to create an efficient process to verify performance equivalence between different code versions. Currently, there is no access to profiling tools adapted for developer usage, which makes the task not efficient.

This document will propose a methodology to tackle this issue. Section 1 contains the motivation for developing this methodology and the objectives proposed for the solution. Section 2 explains the workflow used for obtaining the performance Metrics. Section 3 consists of a use case for the proof of concept developed. The use case describes how to run the script used for the performance analysis and identify possible bottlenecks using the metrics. Section 4 presents the conclusions, a valuation of the objectives accomplished, and possible implementation of the methodology.

1. Introduction and objectives

Earth System Models (ESMs) are 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 these developments, some of the implemented changes create an unintended bottleneck in the model performance. Even though ESM 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.

It's necessary to implement an easier way for developers to profile ESMs, to enable this solution and solve the previously mentioned issues.

For this end, we propose to design a wrapper. A wrapper is a program whose principal purpose is to interconnect other programs or functions. In this case, the wrapper will connect the ESM 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.

Furthermore, to demonstrate the feasibility of the design, we will develop a proof of concept of the wrapper for the NEMO model, one of the most used ocean models among European Earth System Models.

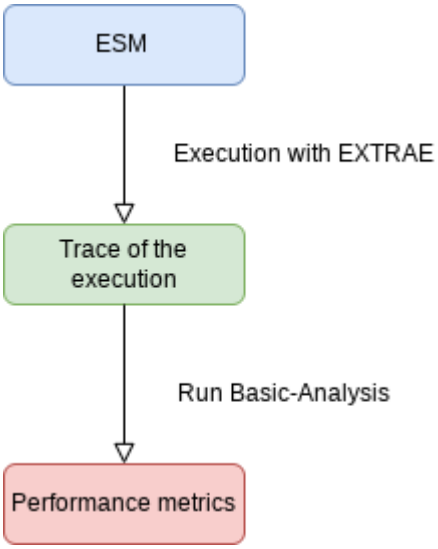

Figure 1: Basic workflow of the elements connected to execute the performance analysis.

2. Methodology

In this section, we will explain the requirements and the general workflow designed to obtain performance metrics from ESMs. Fig. 1 contains the graphical representation of the workflow.

2.1 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

2.2 Workflow

First, we use Extrae, a tool that collects performance information on parallel applications at run time. After executing the ESMs, 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.

3. 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 "***.

	4	24	48	96	192
Global efficiency	94.21	75.01	61.87	56.19	44.46
-- Parallel efficiency	94.21	85.31	81.84	73.75	63.81
-- Load balance	95.59	91.10	90.79	89.40	89.54
-- Communication efficiency	98.56	93.64	90.14	82.50	71.26
-- Serialization efficiency	99.51	97.81	96.65	92.48	88.75
-- Transfer efficiency	99.04	95.74	93.27	89.21	80.29
-- Computation scalability	100.00	87.92	75.61	76.18	69.68
-- IPC scalability	100.00	102.05	96.54	106.01	112.22
-- Instruction scalability	100.00	86.12	78.24	71.79	61.94
-- Frequency scalability	100.00	100.05	100.10	100.10	100.25

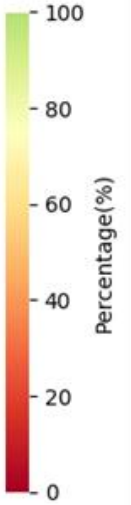

Table 1: efficiency_table-matplot.png generated by the wrapper using ORCA2_ICE_PISCES reference configuration, NEMO output files are disabled.

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 Tab. 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.

4. Conclusions

The wrapper developed; 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 of the compilation of the ESM.
- It's possible to modify the proof of concept to work on different ESM because the core workflow does not depend on NEMO.

The methodology proposed in this report will allow development teams to identify performance bottlenecks without requiring big effort. Moreover, it is easily incorporated into the development process, for example, when requesting a branch merge. In this case, users would only have to compare the metrics and verify that the performance did not decrease before applying changes.