Performance evaluation of the hybrid interactive placement with HIPPO of SCRIP interpolation tasks

E. Maisonnave, A. Piacentini

WN/CGCM/19/94

Table of Contents

Abstract

A library that controls of MPI process placement and OpenMP thread affinity at run-time (HIPPO) is introduced in OASIS and used to perform SCRIP interpolation with the most efficient decomposition, without using environment variable and allowing a different parallelism in the rest of model computations. Our tests shows that performances are preserved and that differentiated parallel layouts are correctly set from SCRIP and further calculations like the NEMO ocean model in its BENCH configuration. Preliminary results suggests that the NEMO code itself could benefit of different parallel layouts.

---

1. Rationale

Computations of the SCRIP interpolation package [1] included in the OASIS coupling library [2] were recently distributed and multi-threaded [3] to improve their performance. Basically, at coupled model initialisation phase, the SCRIP library can be called to calculate the interpolation weights and addresses. These variables will be used to perform the coupling field interpolations needed afterward in the model time loop. During the initial phase, on each computing node, are deployed a predefined number of OpenMP threads, which is supposed to be optimal if corresponding to the total number of physical cores per node. At the same time, every MPI processes but the first of the node are left idle and blocked on a passive MPI barrier. In some cases, the calling coupled model(s) could also require a mixed MPI-OpenMP decomposition, which ratio could be different from the SCRIP one. In this cases, the environment variables that define the number of OpenMP threads, **`OASIS_OMP_NUM_THREADS`** (for SCRIP) and **`OMP_NUM_THREADS`** (for the rest of the code) have to be defined with different values.

This approach leaves some constraints:

- the OpenMP affinity, prescribed via environment variable (**`KMP_AFFINITY`**) is necessarily the same for SCRIP and the rest of the code,
- the passive mode of the MPI barrier needs to be prescribed, in some machines, via an environment variable (**`I_MPI_WAIT_MODE`**). This status of the MPI library cannot be changed at runtime and could severely slow down the MPI exchanges occurring during the model integration,
- possible developments of the OASIS library (dynamic coupling, required by new Lagrangian based definition of the discretisation, see e.g. [4]) would required to call the SCRIP package in the time loop. An alternate calling of code sections with different optimal MPI/OpenMP ratio or OpenMP affinity would be even more crucial for performance.

The recent development of the HIPPO library [5] brings a compact, portable, compiler-independent set of APIs and makes possible a fine and dynamic hybrid parallel definition from inside a C or FORTRAN code. Hosting HIPPO functions in OASIS (a.k.a. "Opération Caravanserail") should ensure the maximum independence of MPI/OpenMP definition for SCRIP routines in one hand and the rest of the code on the other hand. Our first implementation aims to:

- verify that the previous external parallel profile (MPI/OpenMP ratio, MPI and OpenMP environment variables), or *HIPPO layout*, can be redefined at runtime via HIPPO APIs without performance loss,
- check that this layout can be changed after SCRIP calculation to let the model calculations be performed at maximum speed

2. Implementation

To make the HIPPO library available, one would have to also previously install :

- the Portable Hardware Locality (hwloc) software package [6] which provides a

portable abstraction (across OS, versions, architectures, ...) of the hierarchical topology of modern architectures, including NUMA memory nodes, sockets, shared caches, cores and simultaneous multithreading
- the research project [7] which led to the implementation of the hsplit library for hierarchical splitting of MPI communicators.

Both libraries, and HIPPO APIs were quickly compiled on the **beaufix2**[1] Météo-France supercomputer, relying on Intel v16.1.150 and associated v5.1.2.150 MPI library.

Since testing is the only goal of this exercise, and HIPPO an still ongoing project, the comprehensive transformation of all SCRIP interpolations available in OASIS was not necessary. We focus on 1st order conservative (CONSERV) and gaussian (GAUSWGT) options, which exhibit two different and typical scalability profiles.

On a first step, the setup of the HIPPO layout that will be used in the SCRIP routines is performed at the very beginning of the MPI communicator definition (**oasis_init_comp** routine). HIPPO API is called to simultaneously (i) select the number of MPI processes that will be mapped and bound to perform the SCRIP calculations, (ii) define the number of OpenMP threads that will be forked per MPI process and (iii) set the affinity of these OpenMP threads. Notice that at this stage, the MPI OpenMP threads can be forked to all processing units (PU) of the node. If hyperthreading is activated on the machine (which is the case for **beaufix2**) , hyperthreaded calculations can be prescribed here. The layout definition is now internally defined and does not depend on environment variables. However, **KMP_AFFINITY** must be set to "**respect,none**", to let HIPPO modify these values at runtime.

On a second step, just before OpenMP parallel section (SCRIP weight and address calculations, **remap_conserv** & **remap_dist_gaus_wgt** subroutines), we get the threads number as defined previously via HIPPO for SCRIP calculation purpose and organise accordingly the OpenMP parallel loop. At loop start, threads are effectively bound with the appropriate affinity, calling the **HIPPO_Layout_omp_thread_bind** API.

Two toy models, derived from the example available within the OASIS3-MCT v4 we used in this study[2], are adapted to the purpose of the experiment. After the definition phase of the interpolation (performed at initialisation in **oasis_endef** routine), fake calculations are performed and exchanges periodically made between the two toys to simulate a realistic communication pattern between both.

3. Results

We first check that the default behaviour of the SCRIP parallel profile can be reproduced during the "weight and address" calculation phase of the coupled run, without performance loss. On Figure 1a & 1b, we can deduce from the blue and red lines (respectively before and after HIPPO instrumentation, 1 MPI per node, 40 threads per MPI process, alternate affinity) than our modifications do not badly affect SCRIP parallel performances. In the case of the

1 https://www.top500.org/system/178962

2 Directory: **oasis3-mct/examples/test-interpolation**

conservative interpolation (Fig. 1b), we can even notice a small improvement.

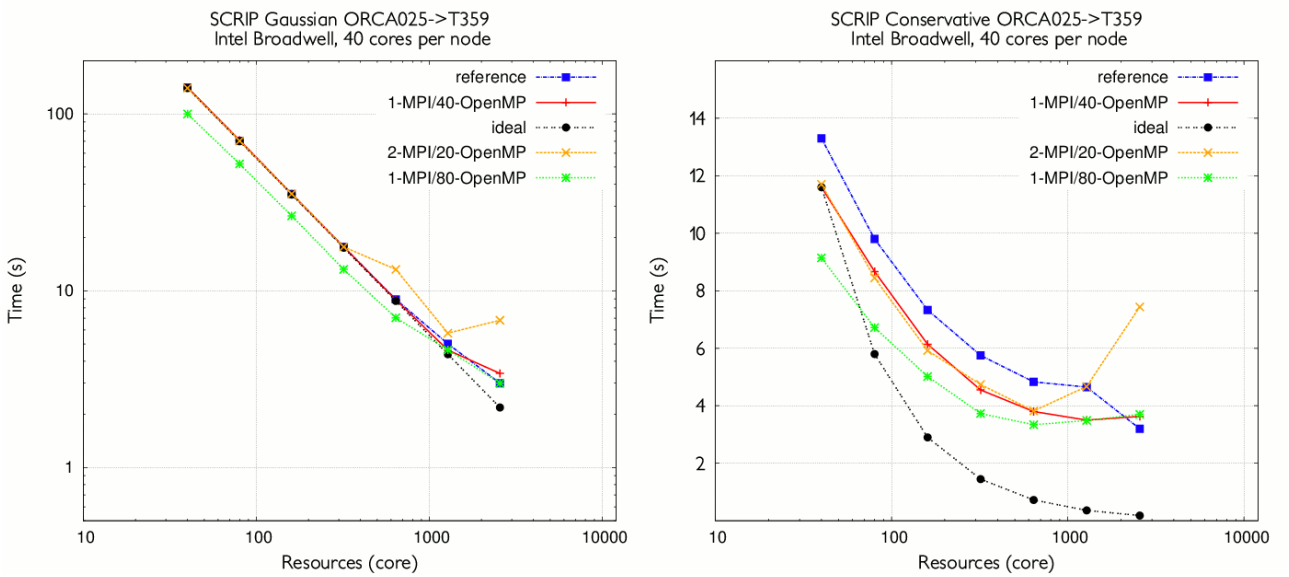

Figure 1: restitution time of SCRIP weight and address computation for gaussian (left) and 1st order conservative (right) interpolation, with various MPI tasks/OpenMP threads ratio, as calculated with and without (blue lines) OASIS3-MCT v4.0 HIPPO-instrumented library

Different ratios of MPI tasks/OpenMP threads lead to different results. The allocation of 2MPI tasks per node downgrades the scalability of both interpolations. At the opposite, the algorithm seems to take benefit of multi-threading (1 task MPI per node, 80 threads) but mainly for the smaller values of node number. Affinity of OpenMP threads (alternate, compact, scatter) has no significant effect (values not shown).

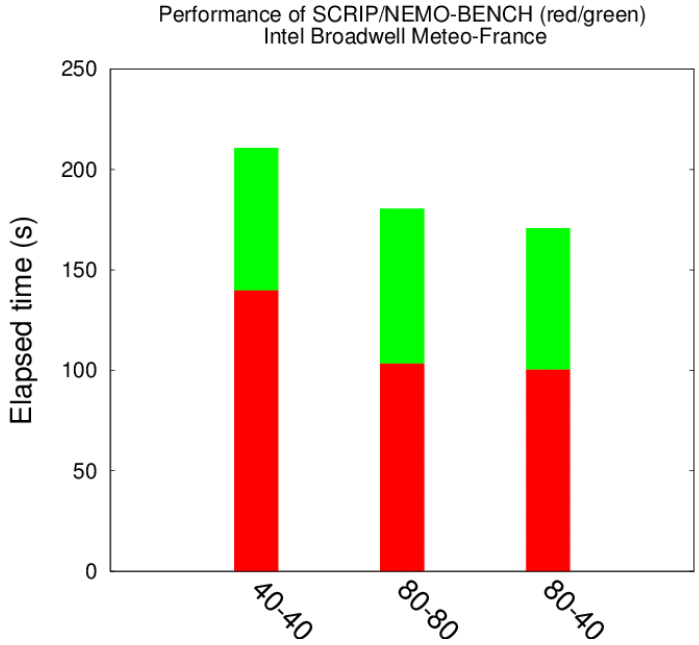

Figure 2: Accumulated restitution time for SCRIP (red) and NEMO-BENCH (green) computations, with different parallel threading (for SCRIP) and MPI decomposition (for NEMO-BENCH)

HIPPO most interesting capability gives the possibility to switch hybrid parallel profile at runtime. To better test this functionality, we introduce larger calculations, the NEMO BENCH configuration [8], in one of our toy models. Our test shows that it is possible to efficiently launch hybrid parallel calculations during SCRIP calculation phase and MPI only calculation during the NEMO BENCH part of the code. This dynamical change of parallelism type is controlled by HIPPO: two different layouts are defined and used sequentially for SCRIP and NEMO routines. Figure 2 shows that the fastest hybrid parallelism is obtained with 80 OpenMP threads, on 40 physical cores, because the compute intensive SCRIP routines can take benefit of hyper-threading. At the opposite, the NEMO-BENCH routines have better performance if only 40 MPI processes are used.

However, a closer look to NEMO internal profiling reveals that some routines (from which the more time consuming, zdf_phy), more compute intensive, can be accelerated by hyper-threading (zdf_phy from 19 to 17.5s), while most of the others are slew down. This suggests that a different decomposition should be used for these two different kind of routines. In particular, an appropriate OpenMP parallelism, only set via HIPPO with the appropriate binding/pining for the compute intensive routines, could enhance the total NEMO performance.

The authors acknowledge the Météo-France machine administration for having facilitated the scalability test. This work has benefited from the IS-ENES3 project, funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 824084

References

[1] Jones, P., 1999: Conservative remapping: First-and second-order conservative remapping, Mon. Weather Rev., 127, 2204–2210
[2] Craig A., Valcke S., Coquart L., 2017: Development and performance of a new version of the OASIS coupler, OASIS3-MCT_3.0, Geosci. Model Dev., 10, pp. 3297-3308, doi:10.5194/gmd-10-3297-2017
[3] Piacentini, A., Maisonnave, E., Jonville, G., Coquart, L. and Valcke, S., 2018: A parallel SCRIP interpolation library for OASIS, Working Note, **WN/CMGC/18/34**, CECI, UMR CERFACS/CNRS No5318, France
[4] Samaké, A., Bouillon, S., Rampal, P., Ólason. E., 2017: Parallel Implementation of a Lagrangian-based Model on an Adaptive Mesh in C++: Application to Sea-Ice. Journal of Computational Physics, dx.doi.org/10.1016/j.jcp.2017.08.055
[5] Piacentini, A., API of the Hybrid Interactive Placement in Palm and Oasis (HIPPO) library, https://nitrox.cerfacs.fr/pae/hippo, restricted access
[6] Broquedis, F., Clet-Ortega, J., Moreaud, S., Furmento, N., Goglin, B., Mercier, G., Thibault, S. and Namyst. R., 2010: hwloc: a Generic Framework for Managing Hardware Affinities in HPC Applications. In *Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP2010)*, Pisa, Italia, February 2010. IEEE Computer Society Press. https://hal.inria.fr/inria-00429889
[7] Goglin, B., Jeannot, E., Mansouri, F., Mercier. G., 2018: A Hierarchical Model to Manage Hardware Topology in MPI Applications. [Research Report] RR-9077, Inria Bordeaux Sud-Ouest; Bordeaux INP; LaBRI - Laboratoire Bordelais de Recherche en Informatique, pp.32. ⟨hal-01538002v6⟩
[8] Maisonnave, E. and Masson, S., 2019: NEMO 4.0 performance: how to identify and reduce unnecessary communications, Technical Report, **TR/CMGC/19/19**, CECI, UMR CERFACS/CNRS No5318, France