EGU22-13484, updated on 13 Aug 2022
https://doi.org/10.5194/egusphere-egu22-13484
EGU General Assembly 2022

Preparing NEMO4.2, the new NEMO modelling framework for the next generation HPCinfrastructures

Francesca Mele[1], Italo Epicoco[1,2], Silvia Mocavero[1], Daley Calvert[3], and Mike Bell[3]
[1]Euro-Mediterranean Centre on Climate Change, Foundation, Italy
[2]University of Salento, Dep. Engineering for Innovation, Lecce, Italy
[3]Met Office, Exeter, UK

Nowadays one of the main challenges in scientific computational field is developing the next generation of HPC technologies, applications and systems towards exascale. This leads to focus the
efforts on the development of a new, efficient, stable and scalable NEMO reference code with improved performances adapted to exploit future HPC technologies in the context of CMEMS systems.
On the main factors that limit the current scalability is an inefficient exploitation of the single node performance. Different technical solutions have been tested to fully exploit memory hierarchies and
hardware peak performance. Between all, the fusion of DO loops together and by dividing the computation over tiles are the two optimization strategies more efficiently take advantage of the cache memory organization. This work focuses on the first one.
The loop fusion is a transformation which takes two adjacent loops that have the same iteration space
traversal and combines their bodies into a single loop. This optimization improves data locality so giving a better exploitation of the cache memory and a reduction of the memory footprint because the
temporary arrays can be replaced with scalar values.
Performance tests have been executed on a domain size of 3002x2002x31 grid points running 1-year
GYRE_PISCES simulations with IO disabled on the Zeus Intel Xeon Gold 6154 machine, available at CMCC. An increasing number of cores - from 504 to 2016 – have been used to test experiments with
the different HPC options.
The analysis focused on the routines where the optimizations have been applied. The use of the extended halo introduces a penalty in the execution time that grows as the number of processes increases and generally the use of loop fusion optimization slightly improves the performance. For many routines, as subdomains get smaller, the improvements due to optimizations are less significant.
The simultaneous application of all optimizations leads to an improvement between 10% and 50%

(except for lateral diffusion). Looking at the total elapsed time, the new HPC optimizations speed up

the elapsed time of a factor 1.25x. Unfortunately, non-optimized routines mitigate this improvement.

The same scalability test has been repeated running 1-month ORCA025 simulations with the output
set to be produced at the end of the run. The results show that the use of loop fusion optimization slightly improves the performance. The use of tiling in ORCA025 introduces less benefits with reference to GYRE. The simultaneous application of all optimizations doesn't lead many benefits in ORCA025 since the improvement concerns only a subset of routines with the Tracer lateral diffusion
routine getting worse in all cases.
In conclusion the impact of the new optimized code behaves differently depending on the configuration. The overhead introduced by the extended halo implies a computation time cost that the
proposed optimizations are able to regain difficultly. Tiling is the aspect with the highest impact in these optimizations (especially in GYRE) and loop fusion has in general a low impact. The optimizations
should be applied to all the rest of the code to obtain more benefits.