Optimization of an ocean model using performance tools

Oriol Tintó Prims[1,3], Miguel Castrillo[1], Harald Servat[1], Germán Llort[1], Kim Serradell[1], Oriol Mula-Valls[2] and Francisco Doblas-Reyes[1,2,4]

[1]Barcelona Supercomputing Center, [2]Institut Català de Ciències del Clima, [3]Universitat Autònoma de Barcelona, [4]Institució Catalana de Recerca i Estudis Avançats **Contact: oriol.tinto@bsc.es**

The NEMO model consumes billions of computing hours every year... Using a performance analysis methodology we boosted the performance saving millions of computing hours and energy

What is NEMO?

NEMO is an ocean global circulation model that includes several sub-models:

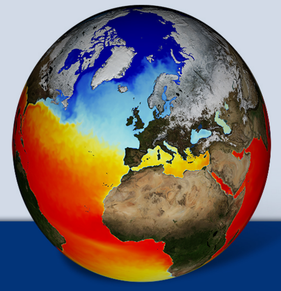
- **OPA**, its nucleus, solves the dynamics and thermodynamics of the ocean.
- **LIM**, a sea-ice model.
- **TOP**, a biochemistry model.

Why is NEMO important?

NEMO is developed by an European Consortium and used by hundreds of institutions.

In the CMIP5 project 5 of the 28 Earth system models used included NEMO as an oceanic engine and the resources needed exceeded a few **billion** computing hours.

Visit the NEMO Website



Visit the BSC Tools Website

BSC Tools

Performance analysis tools allow application developers to identify and characterize the inefficiencies that caused a poor performance. In weather and climate applications this kind of tools are used very sparsely.

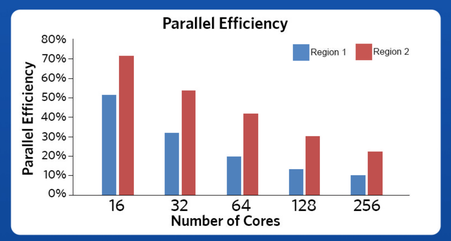
- **Paraver:** A very powerful performance visualization and analysis tool based on traces that can be used to analyze any information that is expressed on its input trace format.
- **Dimemas:** Simulation tool for the parametric analysis of the behavior of message-passing applications on a configurable parallel platform.

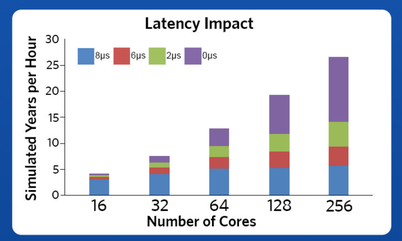
Analysis

In the figure above, function timelines corresponding to different core number configurations are compared to reveal where the scalability issues come from and how the different function times evolve.

The sea-ice horizontal diffusion routine and the ocean surface pressure gradient solver are revealed to be the main bottlenecks. From now on we will refer to these regions as Region 1, for the sea-ice model, and Region 2, for the ocean surface pressure gradient.

BOTTLENECK DISSECTION





Cluster tracking techniques have been used to identify different computational trends in the regions of interest and then assess their scaling behaviour. With such technique we learned that while there is a decrease of computational efficiency, mainly related with code duplication, only a very small part of the overall efficiency loss can be attributed to computation.

As it can be seen in the left above figure, communications are the main performance problem since even in the 16-Core case the parallel efficiency is bad and it decreases dramatically when increasing the number of processors. Moreover, the figure above right shows how sensitive the model is to network latency, supporting the idea that there is a problem in the communication.

With both computation and communication analysis of the model bottlenecks we can conclude that although both elements present a loss of efficiency when scaling the model, the part corresponding to communication issues is much more important. This is true for two reasons: because in the 16-Core case the parallel efficiency is already bad and also because it drops faster than the computational efficiency.

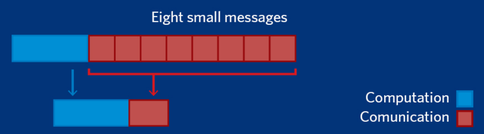
Environment

Marenostrum III at Barcelona Supercomputing Center

- 16 CPU cores per node (2 x Intel SandyBridge-EP 8-core at 2.6 GH)
- 32GB main memory per node (8 x 4 GB DDR3-1600 DIMMS)
- 3,056 compute nodes, 48,896 processor cores
- Infiniband FDR10, Linux - SuSe Distribution

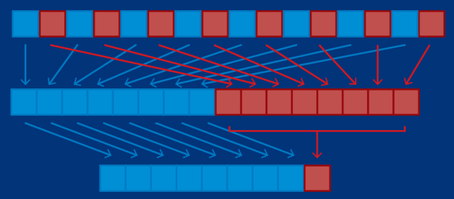
Optimizations

MESSAGE PACKING



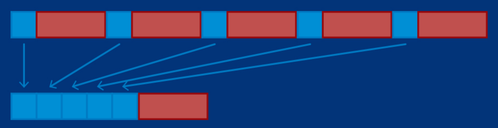
Taking in account that NEMO is much more sensitive to latency than to bandwidth, the aggregation of messages is the best way to reduce the time invested in communications. Therefore, consecutive messages can be packed wherever the computational dependencies allow to do so. This approach can be applied both to point to point and collective communication.

REORDERING



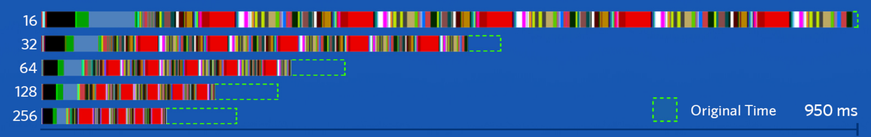
In order to apply the message packing optimization to as many routines as possible, it was necessary to rearrange some computation and communication regions, taking into account the dependencies between them, to reduce the number of messages.

CONVERGENCE CHECK REDUCTION



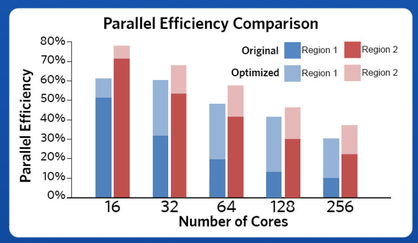
Some routines use collective communications to perform a convergence check in iterative solvers. The cost of this verifications is really high, reaching a 66% of the time in some cases. Wherever the model allowed it, we reduced the frequency of this verifications in order to increase the parallel efficiency.

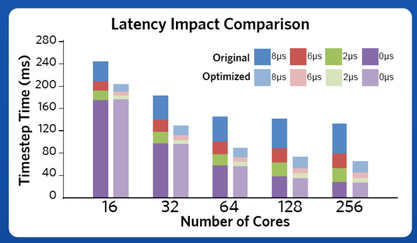
Results



The trace visualization of the NEMO model after applying the optimizations, shown above, allows the comparison with the traces (shaded with green dashed lines) of the original version. This comparison clearly illustrates where the optimization has had an impact, which is obvious even with a 16-core configuration.

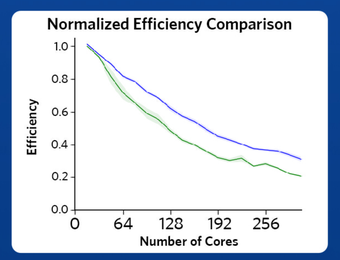
COMMUNICATION IMPROVED

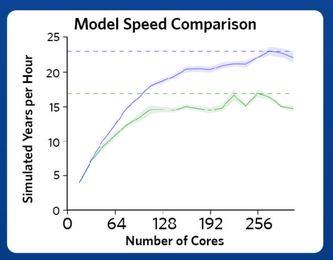




By applying these three optimizations we reduced substantially the quantity of messages that NEMO interchanges. The figure above left shows that these optimizations improved substantially the parallel efficiency, especially in Region 2, while the figure above right shows how the optimizations reduced the latency impact on the model, making the duration of the time-step less sensitive to the network.

SCALABILITY IMPROVED





While the original version of NEMO can achieve a simulation speed of 16.8 simulated years per hour in the best case, our optimized version rises up to **23 simulated years per hour**. This means that the maximum simulated speed that NEMO can achieve has **increased by 37%**. The increase in efficiency allows to simulate, in the 128-Core case, **22% more years** when using the optimized version.

With a detailed performance analysis we could identify the bottlenecks which constrain the scalability of the NEMO ocean model and then find **straightforward optimizations** that allowed us to **improve substantially** its **performance and efficiency**.