A resiliency strategy for NEMO Ocean Model

Eric Maisonnave

TR/CMGC/12/13

Abstract. Towards Exascale Climate Model implementations, we propose to explore the possibility of resilience on the NEMO ocean model, without the help of spare resources and avoiding to handle the associated on-line data recovery. A simple, non intrusive error repair have been implemented and tested, simulating failure on one core and analyzing its consequences on the corresponding sub-domain of the global ORCA2 grid. Despite locally restrained biases, a simulation subject to a one month long failure on a Pacific Ocean sub-domain has been achieved without performance losses. We are encouraged to evaluate with larger configurations our strategy impact on global circulation.

Keywords: Fault Tolerance, Ocean Modeling, Numerical Algorithms for CS&E, Multiscale and Multiphysics Problems

1 Introduction

Exascale is a major challenge for climate modeling. For decades, climate modeling has been benefiting of computing progress; now, to fit supercomputing hardware and software evolution, legacy codes, made of millions of lines, have to be adapted or partly re-written for more efficiency.

Progress have been made in parallelism: changing spatial discretization, new dynamical cores avoid costly filtering at pole regions [1]; better organizing data throughput, new scientific libraries reduce important bottlenecks [2].

Supercomputer new facilities are fully exploited to increase model resolution, better represent low scale phenomena and understand scale interactions [3]. Most of the Top500 machines have been, and presently are, operated by climate modeling groups all over the world (Earth Simulator in Japan, Oakridge or Argonne machines in the United States, PRACE tier-0 facilities in Europe ...)

In this context, given that machine gathering millions to trillions of computing cores is a clear target of climate modeling community, it appeared that the emerging issue of fault tolerance has to be taken into account. At the origin of our work, the french ANR collaborative project “SPADES” [4], involving both climate modeling engineers and resiliency experts, has been set up to prepare climate models to fault tolerance.

Obviously, an efficient resiliency strategy could not only rely on future hardware and software environment enhancements: our parallel MPI-based climate models

have to be adapted to be able to handle fault tolerant MPI implementations [5], assuming that MPI will be enhanced [6] to satisfy Exascale constraints [7].

To make a program resilient is a non transparent change: fault tolerance handling leads to modify scientific programs. We assume that MPI will detect and provide information on failure characteristics but, at this point, model modifications will be necessary to start the repair process and/or contain fault propagation.

2 Purpose

In this study, we propose to describe a strategy that will lead to implement a resilient climate model component (ocean), detail this first implementation and give a preliminary result. This practical work comes up against different difficulties.

Even if standards are still a work-in-progress, the different possible behaviors of fault tolerant MPI parallel library have to be foreseen. Indeed, the message passing system is a key tool to articulate a resiliency strategy on scientific programs such as climate models: it is when MPI communications are necessary to keep calculating on a MPI process that failure can be detected by all instances and that a repair process can be started.

The structure of repair process strongly depends on the kind of preparatory fault detection and handling provided by the new MPI implementations, as well as their different performances. They could be suitable or not for the specificity of climate modeling. One of the challenges on resiliency strategy efficiency lies on this choice: a high degree of fault tolerance (full data recovery, same computing resources after than before failure) requiring too much time (comparable to simulation duration) cannot be considered as efficient.

Consequently, as a first step, hypothesis on the best suitable MPI error handling have to be made, considering both state-of-the-art standard of fault tolerant MPI and version of highest parallel climate models. In order to validate some of those hypothesis, a repair process has been implemented on a selected climate system module: it will be described in the present work.

But an effective and *realistic* test of a resilient version of our model requires both hardware (Exascale machine) and software (massively parallel climate model) environment that will not be achievable prior to several years. Consequently, on a first step, we will choose an existing climate model configuration and mimic conditions that will prevail on Exascale machine. Our first objective is not to quantitatively validate a fully resilient version of a climate model, but to confirm the first hypothesis on which our implementation strategy is based, that repair process

duration can be achieved on a reasonable time and that resulting errors do not lead to numerical scheme divergence.

More generally speaking, we expect that this experiment will give us a clearer idea on how to design the future version of climate models (massively parallel, using graphics processors …) that will be developed in the coming years, taking into account resiliency requirements. This work would be strongly facilitated if an error handling strategy, simple, efficient and compatible with climate modeling characteristics, has been already tested.

3 Method

3.1 Climate model and MPI state-of-the-art

As far as we know, there is no existing full or partial climate model implementation that could be considered as resilient. We call full climate model the complete system necessary to simulate the Earth Climate with enough complexity to address scientific questions such as Climate Change or seasonal to decadal forecasts or predictions. CGCMs (Coupled Global Circulation Models), encompassing different modules gridded on the whole globe, are mandatory for such experiments. Those components can be assembled on a single executable [8] or launched in parallel executables, coupled with an appropriate tool [9].

In order to simplify our problem, we decide to choose one of the two essential components of the ARPEGE-NEMO CGCM, widely used on the most advanced supercomputers [10]. Indeed, NEMO Ocean model [11] combines standard characteristics of climate model components such as Fortran writing, MPI parallelism following 2D spatial decomposition. Due to its long elapsed duration (from days to months), an ocean only simulation is performed as a climate one: its 3D prognostic variables have to be regularly checkpointed (every simulated month to every simulated year) and simulation restarted from this point.

Moreover, the NEMO code length is compatible with a quick adaptation for an error handling; its MPI interface is isolated on a single file of only 23 routines.

The multi-executable structure of our CGCM allows separate adaptation of a single module. The fault tolerant NEMO version will be used on stand alone mode. It must be clear that, to fully evaluate performances and reliability of a climate model, the other modules and the coupler should also be made resilient. But modularity of the system should facilitate incremental modifications.

NEMO model has a 3D spatial discretization on the whole globe (spherical coordinates). Parallelism consists on separating grid points into sub-domains (following latitude and longitude axes). The MPI library is used to exchange information at boundaries of the sub-domains, mostly through point-to-point communications, several time per time step, for several variables.

We assume that future implementations of resilient MPI will detect and begin to handle failures at communicator level and let the application manage the repair [12]. We suppose that this implementation will be mostly similar to those previously developed on FT-MPI / OpenMPI. Our practical test will be made with OpenMPI.

3.2 Error handling strategies

Error handling begins on NEMO code when an error is signaled by any MPI library call. We have identified 3 different error repair strategies ensuing 3 different MPI error handling type:

(a) No repair process, MPI is disabled. It implies that model has to save current prognostic variables, stop and restart.
(b) The failed process is replaced by a new resource. It implies that the model has to recover the lost part of its prognostic variables and resume simulation on the corresponding time step.
(c) The failed process is not replaced but point-to-point communications are still possible except with failed process, and a special treatment is done for collective communications. It implies that simulation can go on, but with missing information.

A simple analysis of NEMO present behavior on scalar machines already reveals how expensive checkpoint/restart operations are. Their cost is supposed to increase [13] with resolution (Exascale computing will address problems of more and more accuracy) and parallelism (parallel access to disk or cache memory). Climate modeling is already considered as one of the most consuming sciences regarding to memory and output requirements: to reduce those needs and satisfy future Exascale constraints, we think that error treatment only based on checkpointing has to be avoided: perform a checkpoint/restart operation as often as failure frequency would lead system to spend most of its time in I/O operations rather than calculating.

The same data volume considerations suggest us to prefer not to deal with data recovery. This operation implies, on climate model, to continuously keep a copy of an important volume of data, certainly uneasy to transfer to the spare memory resource during error repair. Moreover, this additional cost does not include the additional time needed by MPI management to dynamically reallocate an equally efficient spare resource.

The error handling strategy we propose is based on the following assumption: when a failure occurs, a simulation can be carried out despite missing data and calculations on the sub-domain corresponding to the failed resource.

A clear consequence is the simulation non reproducibility. Moreover, if data located on the failed sub-domain are lost, the model no more ensures energy and mass conservation.

This bias will have to be evaluated and compared to other sources of non conservation. The main argument that leads us to move forward in this direction is that increasing parallelism leads to decrease the relative failed sub-domain size, compared to the global domain. The question is then: at which parallelism level could the bias and its propagation be considered as a simple perturbation ? Even though this limit can not be found on present machines, a test case configuration of our model has been chosen to represent the problem as realistically as possible and give a first idea on the acceptable limit.

To reduce the error impact, our error repair strategy combines three repair levels, which first two will be detailed at 'Implementation and first test' chapter:

- Just after failure, no spare resource is needed; model keep exchanging boundary conditions at sub-domain limits through the failed sub-domain, using values of the nearest valid sub-domain neighbor.
- Before resuming calculations, area of grid point neighbors (halo) to the failed sub-domain is increased to compensate the missing grid points area.
- When the next regular checkpoint is performed, values of those neighbors grid points are extrapolated to fill failed sub-domain on restart variables. Calculations can then be started again with regular resources number.

3.3 Test case configuration

Several global or regional NEMO configurations are currently available on supercomputers, but resolutions expected to be used at Exascale are relatively difficult to operate on present machines.

On the basis of the last considerations on the topic [14], we guess that a code able to fully exploit an Exascale computer should be parallelized on a trillion of cores. It is presently impossible to ensure that CGCMs could reach such level. Considering present performances, a million cores hypothesis seems more reasonable.

On these machines, a node could be composed of 10^3 to 10^4 cores. It is still unclear if the entire node will be effected entirely of partially by failures. Consequently, the ratio between the global Earth surface area and the area affected by failure could vary a lot depending on the various hypothesis.

It seems presently impossible to determine a realistic value of this global/sub-domain ratio, necessary to evaluate realistically the impact of the error repair strategy on ocean physics. Nevertheless, the error repair algorithm can be implemented on NEMO and tested with an existing configuration such as ORCA2 (2 degrees resolution, global domain) on 100 cores of CINES SGI Altix supercomputer. For a failure of one sub-domain, this ratio is equal to 100.

Even though we suppose that reaching an Exascale-realistic value of this ratio requires finer model resolution and higher parallelism, the present configuration will help you to quickly test if, with the implemented error handling, and despite failure affecting a non negligible part of its global domain, the model is able to go on and process the entire simulation.

Failure simulator are not already available on our supercomputers, as well as MPI implementation that can provide error messages after failure detecting. Our implementation will have to simulate those events with simple Fortran instructions.

3.4 Implementation and first test

Our error repair routine (mpp_reinit) follows the existing structure of the initial global domain partitioning (mpp_init). In this routine, index of neighbors sub-domain are initialized. When failure occurs, this routine is called to redefine sub-domain neighbors and redirect communications through the failed sub-domain.

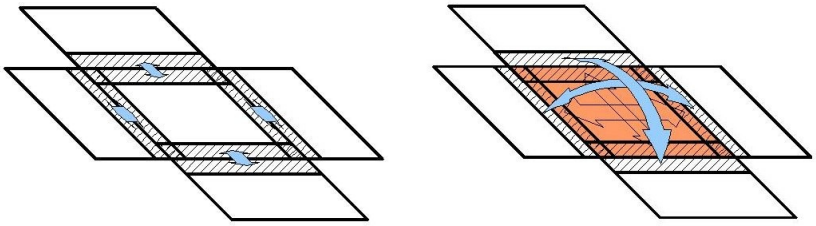

Fig. 1. First level of implemented error repair strategy. Shaded arrows represent MPI exchanges necessary to fill extra lines/columns (hatched) at boundaries of sub-domains. At left, a regular situation. At right, after failure of central sub-domain (shaded area). Transparent arrows represent inner communications on pseudo-failed sub-domain.

In this new configuration, south-northernmost (east-westernmost) grid points neighbors of the failed sub-domain, located on boundary lines (columns) are spaced at a sub-domain width (length) interval.

To temporary simulate failures, which can not be produced by present hardware and handled by the available MPI library, the failed sub-domain calculations must be stopped (Fig 1, right, shaded area). This could be done replacing ocean grid points by land grid points. As communications from boundary are still necessary on the sub-domain to avoid MPI collective blocking, eastern-western (northern-southern) boundary conditions are internally exchanged (Fig 1, right, transparent arrows). An advantage of this solution is that collective communications can go on: for example, global mean values can be calculated, via a MPI collective communication, with a global domain, pseudo-truncated (masked) on the failed sub-domain.

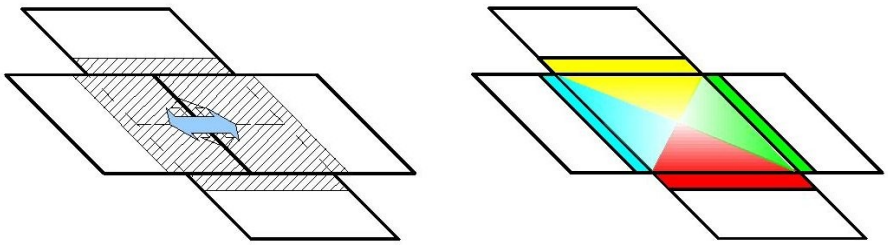

Fig. 2. Second and third implemented error repair strategy. Width (or height) of grid point neighbors of the failed sub-domain are increased to compensate missing area area (right). During the next regular checkpoint/restart phase, the restart file is repaired: missing grid point values are extrapolated from values of grid point neighbors. Simulation can be resumed on the whole domain (left).

Both to ensure gradient calculation validity on those regions and conserve total area of the global domain, length/width of those grid points must be increased to half failed sub-domain length/width (Fig 2, left). One can consider that resolution of grid point located at failed sub-domain boundaries has been increased until the missing area was filled.

On our first experiment, a failure is simulated on one sub-domain located at center of Pacific Ocean, to avoid a special treatment necessary if its boundaries intersect continents. The experiment starts from the result of a previous regular one.

In the worst case, our first and second level error repair must be applied during a one month period, from the beginning of a run to the first regular checkpoint: a one month long simulation has been performed successfully in these conditions, validating stability of the implemented repair algorithm.

Results of a regular and a fault tolerant simulation have been compared. Differences are maximum at sub-surface, where currents maxima are observed and where the damping effect of surface fluxes is weaker. After a short increase following failure, their values tend to stabilize. Spatially, differences decrease under model

variability level at a distance equivalent to one failed sub-domain length. Those first analysis suggests that the implemented error repair and handling leads to a locally restrained and temporally constant bias.

No significative slowing-down has been measured during or after repairing. For example, the number of iterations to reach convergence on pressure solver regains its initial value after a short period of perturbation.

4 Conclusion

The present study excludes impact evaluation of the third repair level of our strategy, after model prognostics values at boundary of the failed sub-domain were extrapolated on missing grid points during simulation restart (Fig 3, right). Resuming calculations on the failed sub-domain is supposed to nullify the observed biases after a given time period.

A statistical evaluation of the effect produced by our full error repair strategy must be performed. It will be necessary to produce ensembles of regular and fault tolerant simulations, varying failure localization and duration. To try to determine at which parallelism level the error produced by our resiliency strategy could be compared to a simple perturbation, a finer and more realistic ocean model configuration is required, like, for example, NEMO ORCA12 (1/12 degrees, global domain), already available on about 10^5 cores of PRACE tier-0 class machines. In parallel, error severity could be quantified regarding of geographical position of the failed sub-domain.

It is only at this stage that it will be possible to conclude to the possibility of resiliency on the NEMO model, without the help of spare resources and associated on-line data recovery handling. Nevertheless, community must be warned about the non reproducibility and non conservativeness of this solution and probably keeps its use to particular short term simulation such as seasonal forecast or decadal prediction.

Acknowledgments
The authors would like to thanks Ala Rezmerita, Sophie Valcke, Luc Giraud, Olivier Marti, Thomas Hérault and George Bosilca for stimulating discussions about this work. This work has been supported in part by the ANR project SPADES (08-ANR-SEGI-025). It was granted access to the HPC resources of CCRT and CINES under allocations 2011016028 and 2012016028 made by GENCI (Grand Equipement National de Calcul Intensif).

References

1. Satoh, M., Matsuno, T., Tomita, H., Miura, H., Nasuno, T., Iga, S.: Nonhydrostatic icosahedral atmospheric model (NICAM) for global cloud resolving simulations, Journal of Computational Physics, Volume 227, Issue 7, 20, pp. 3486-3514 (2008)
2. Li, J., Keng Liao, W., Choudhary, A., Ross, R., Thakur, R., Gropp, W., Latham, R., Siegel, A., Gallagher, B., Zingale, M.: Parallel netcdf: A high-performance scientific i/o interface. SC Conference, 0:39 (2003)
3. Lévy, M., Klein, P., Tréguier, A.-M., Iovino, D., Madec, G., Masson, S., Takahashi, K.: Modifications of gyre circulation by sub-mesoscale physics, Ocean Modelling, Volume 34, Issues 1-2, pp. 1-15 (2010)
4. ANR SPADES project, http://graal.ens-lyon.fr/SPADES
5. Gropp, W., Lusk, E.: Fault tolerance in MPI programs. Special issue of the Journal High Performance Computing Applications (IJHPCA) 18(3): pp 363–372 (2004)
6. Bosilca, G., Herault, T., Rezmerita, A., Dongarra, J.: On Scalability for MPI Runtime Systems, Proceedings of the IEEE Cluster 2011 International Conference(Cluster'11), Austin, TX, USA (2011)
7. Balaji, P., Buntinas, D., Goodell, D., Gropp, W., Kumar, S., Lusk, E. L., Thakur, R., Traff, J. L.: MPI on a million processors. In PVM/MPI, pp 20–30 (2009)
8. Gent, P. R., and Coauthors: The Community Climate System Model Version 4. *J. Climate*, 24, 4973--4991 (2001)
9. Valcke, S.: OASIS3 User Guide (prism_2-5). CERFACS Technical Report TR/CMGC/06/73, PRISM Report No 3, Toulouse, France. 60 pp (2006)
10. Maisonnave, E., Coquart, L. and Valcke, S.: PRACE preparatory access for high resolution ARPEGE-NEMO porting on Bullx TGCC platform ,Working Note, WN/CMGC/11/77, SUC au CERFACS, URA CERFACS/CNRS No1875, France (2011)
11. Madec G.: "NEMO ocean engine". Note du Pôle de modélisation, Institut Pierre-Simon Laplace (IPSL), France, No 27 ISSN No 1288-1619 (2008)
12. Fagg, G. E., Dongarra, J.:FT-MPI: Fault tolerant MPI, supporting dynamic applications in a dynamic world. In Proc. of Euro PVM/MPI 2000, pp. 346–353. Lecture Notes in Computer Science 1908, Springer, (2000)
13. Schroeder, B., Gibson, G. A.: Understanding Failures in Petascale Computers, J. of Physics: Conference Series, 78, doi:10.1088/1742-6596/78/1/012022, (2007)
14. Dongarra, J., Beckman, P. et al.: “The International Exascale Software Roadmap," Volume 25, Number 1, International Journal of High Performance Computer Applications, ISSN 1094-3420 (2011)