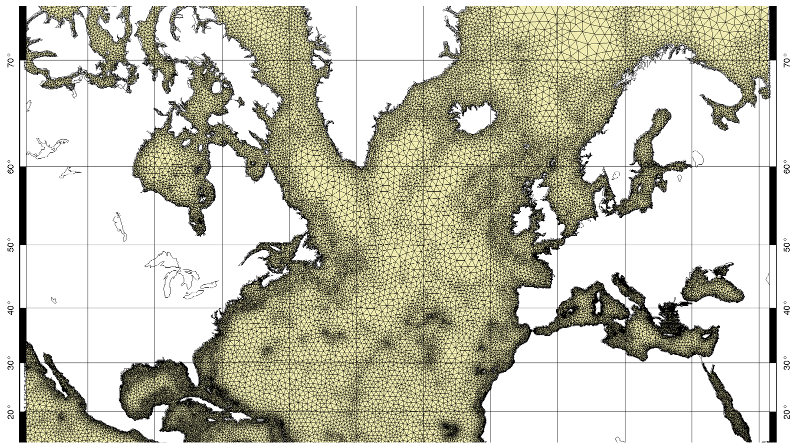
A comparative performance analysis of oceanic numerical models: numerical and computational aspects

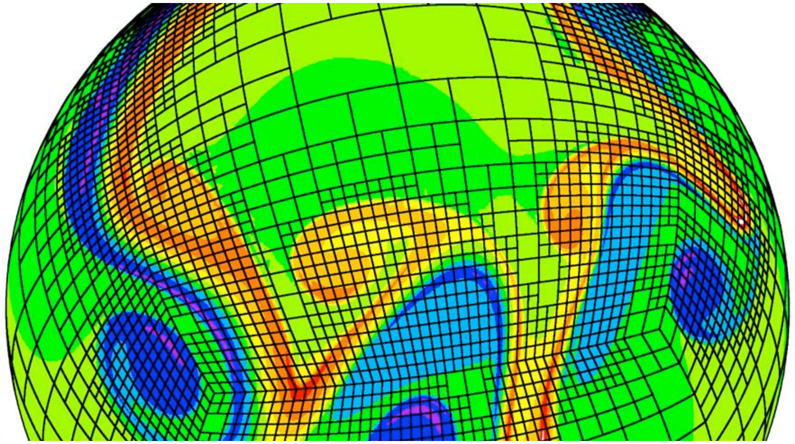
Roland Patoum, Laurent Debreu, Florian Lemarié

Inria and Laboratoire Jean Kuntzmann, Grenoble, France

Performance of oceanic models

- Numerical schemes
- Computational implementation
- Mesh Refinement





NEMO CROCO MARS3D - Main characteristics

- NEMO
  - Low order time stepping for advection (second order for internal waves)
  - Explicit free surface (linear here)
  - Use pointers as work arrays
  - Array order (I,j,k)
  - Masked loops
- CROCO
  - High order time stepping for advection (second order for internal waves)
  - Non linear explicit free surface
  - Smart treatment of work arrays
  - Array order (I,j,k)
  - Masked loops
- MARS3D
  - High order (space-time) time stepping for advection (second order for internal waves)
  - Non linear Implicit (ADI) free surface
  - Array order (k,I,j)
  - Restricted (sea only) loops

Numerical methods of three oceanic models: stability constraints

Computational performance = efficiency of the time integration schemes

	NEMO	CROCO	MARS3D
Internal Gravity Waves	0.46	0.85	2
External Gravity Waves	0.46	0.9	∞
Advection	0.46	0.87	1

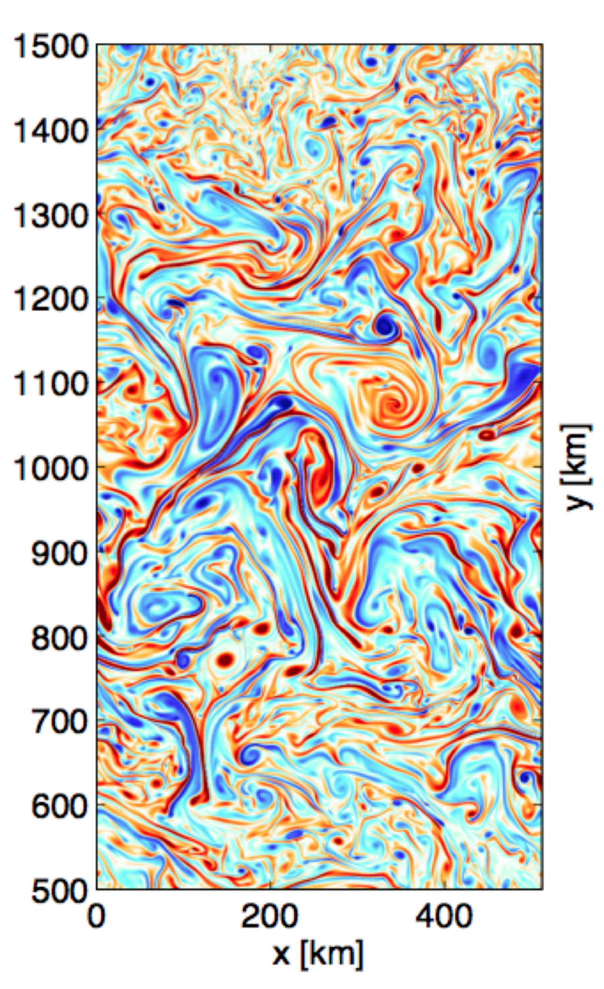
Maximum Courant numbers, Lemarié et al, 2015 (Ocean Modelling)

A baroclinic test case at 2km (Soufflet et al, OM, 2016) 1000 x 250 x 100 grid points

$$\Delta t \sqrt{(c_1)^2_{i,j}\left(\frac{1}{(\Delta x_{i,j})^2} + \frac{1}{(\Delta y_{i,j})^2}\right)} \leq \alpha^{\star}_{\mathrm{igw}}$$

- Maximum time steps

	NEMO	CROCO	MARS3D
$\Delta t(s)$	200	340	320
$\Delta t_{2\mathrm{D}}(s)$	3	6,1	320
$\alpha^{\star}_{\mathrm{igw}}$	0.46	0.85	2

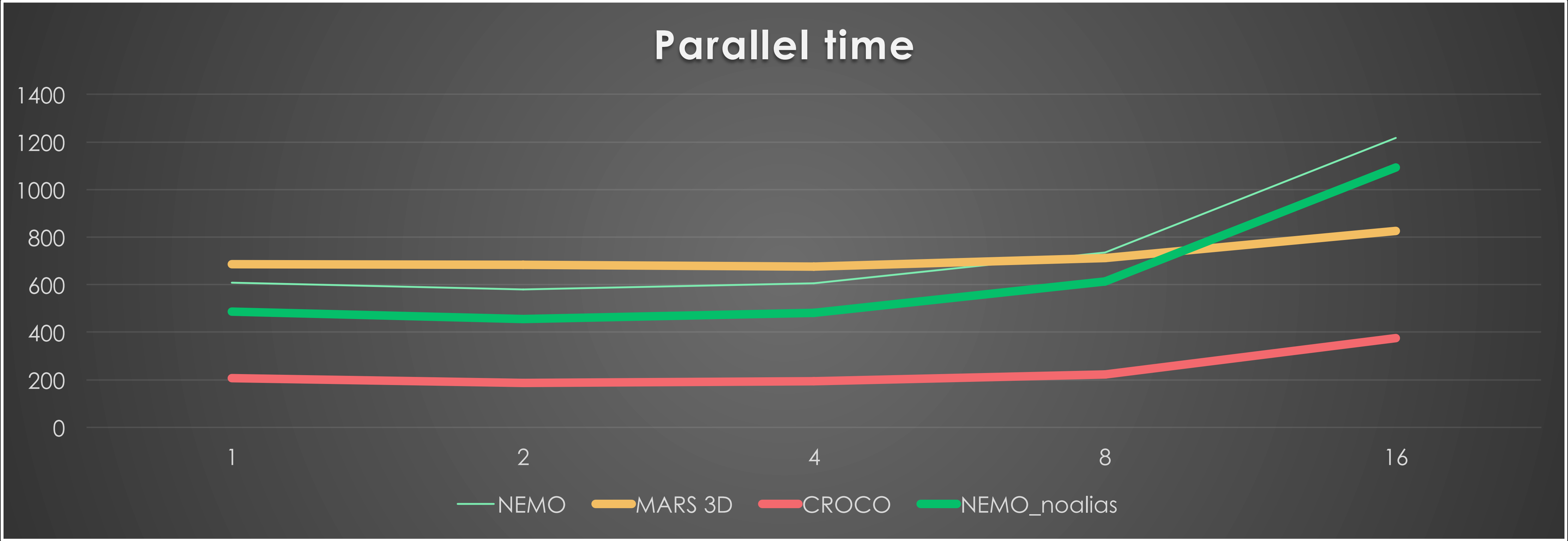


Sequential performance (Intel VTune)

	NEMO	NEMO (noalias)	CROCO	CROCO (autotiling)	MARS
Memory size	2 Gb	2 Gb	1 Gb	800Mb	1,4 Gb
Number of instructions	5,50 Mds	4,3 Mds	3,2 Mds	3,3 Mds	13,9 Mds
Vectorization (%)	40	65	80	78	45
Cache bound (%)	14	10	13	14	71
DRAM bound (%)	31	36	26	21	15
Execution time (s)	609	486	186	160	686

NEMO uses a linear free surface
MARS makes a lot of unnecessary tests on water depth
All codes advect salinity even if not needed …

Parallel performance



Analysis

- NEMO
  - Improvement of time integration algorithm is required
  - Management of work arrays
  - Memory size (unnecessary global arrays ?)
- CROCO
  - Improvement of time integration algorithm is possible
- MARS3D
  - Possible instability of momentum advection

Conclusions

- Improving numerical methods is the easiest way to improve code efficiency
  - (unuseful to try to computationally optimize a code with bad numerics)
  - Need to follow the code evolution
    - Wetting and Drying
    - Non hydrostatic
    - …

→

- Increase of resolution
- Incorporate new physics
- Increase of the size of an ensemble

Other HPC key ingredients

- (Adaptive) mesh refinement
- Online Diagnostics / Visualization
- Hybrid parallelization