OpenMP-based parallel calculations and application to DeepLines, a dynamic underwater lines simulation code

Arthur TALPAERT, Boniface NKONGA

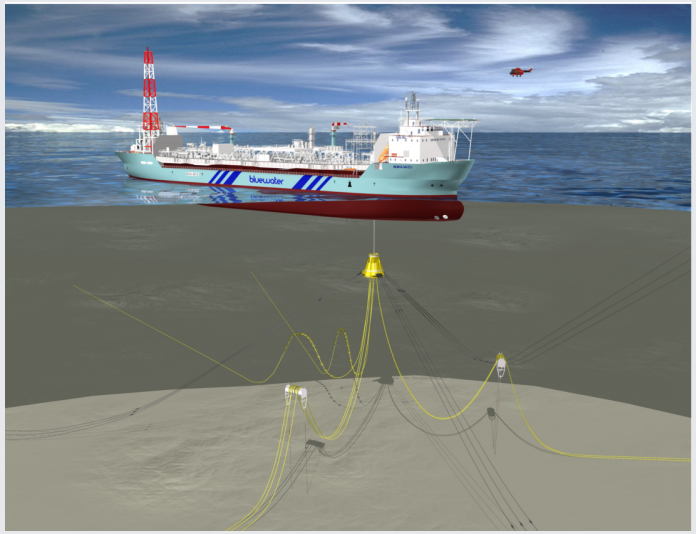
CEA & École Polytechnique, INRIA & Univ. de Nice Sophia-Antipolis

CEMRACS 2014 at Marseille, France

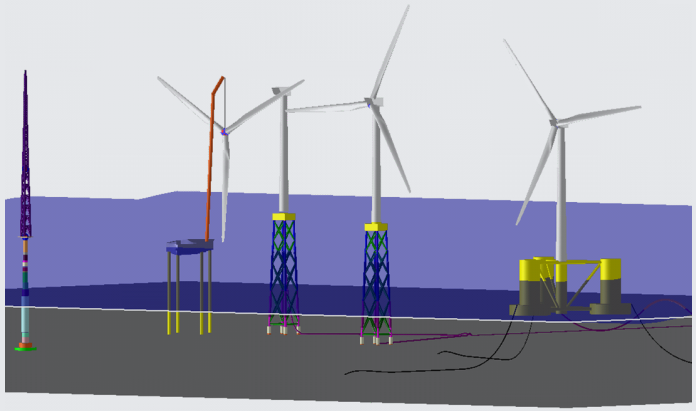
2014-08-27

Motivation

DeepLines, developped by Principia



Bluewater – Aoka Mizu FPSO



Principia – Windturbines on riser

Sometimes hours of computation time $\Rightarrow$ How to improve the efficiency?

Initial situation

Sequential calculation

Mechanics problem equivalent to

$$M_2\ddot{x} + M_1\dot{x} + M_0 x = \lambda \tag{1}$$

After discretization with the Finite Elements Method:

$$Ax(t_{n+1}) = b \tag{2}$$

Parallelization strategy

Assembly of A and b

- Every element `i` undergoes its own influence.
- OpenMP is a good choice because it requires only a small modification of the original code.

```
subroutine pb_construction(nbelmts, A, b)
    !$ use omp_lib
    !<here variable declarations>

    !$omp parrallel do shared (A, b) private(contrib_i)
    do i = 1, nbelmts:
        call contribution(i, contrib_i)
        call assembly(contrib_i, A, b)
    end do
    !$omp end parallel do
end subroutine pb_construction
```

Issues

Competition of threads while writing inside A and b

- Use the `!$omp critical`, `!$omp end critical` to protect the assembly.
- Consequence: a little bit of extra (overhead) computation time. Not the major issue.

Issues

Extensive use of modules with static (global) variables

```
subroutine contribution(i, contrib_i)
   use characteristics
   ! No declaration here of depth.
   depth = readfromfile(i)
   contrib_i = contrib_from_depth(depth)
end subroutine contribution

module characteristics
    real :: depth
    save
end module characteristics
```

Which variables should be shared or private? Make use of `!$ copyin(depth)` and of `!$omp threadprivate(depth)`.

Results

Objective: improve computation time

- Did not reach objective: code still converges but computation time increased. No way to tell whether the converged result is the right one (probably not).
- But path is rather clear for future improvement:
  - using experience of the code's modules
  - using additional tools such as Valgrind (not possible on the machine I was given)

References and funding

Jalel CHERQUI, Pierre-François LAVALLÉE

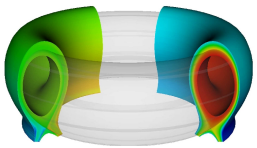
OpenMP, Parallélisation multitâches pour machines à mémoire partagée

IDRIS course material, Nov. 2008

CHANDAR, DAGUM, KOHR, MAYDAN, MCDONALD, MENON

Parallel Programming in OpenMP

Morgan Kaufmann Publishers, Academic Press, 2001



PRINCIPIA

cea

ÉCOLE POLYTECHNIQUE UNIVERSITÉ PARIS-SACLAY

DGA

Inria INVENTEURS DU MONDE NUMÉRIQUE

Université Nice Sophia Antipolis