Programmation parallèle hybride

Marc Tajchman

CEA - DEN/DM2S/STMF/LMES

10/12/2020

Cas envisagés

Du fait qu'il y a souvent plusieurs dispositifs matériels (clusters, machines multi-cœurs, GPU, etc.) et plusieurs outils logiciels associés, il est intéressant d'essayer de combiner leur utilisation.

En fonction de ce qui est disponible sur les machines

1. clusters (plus généralement, machines parallèles multi-nœuds) : MPI,
2. processeurs multi-cœurs : OpenMP (et autres outils : TBB, std::threads, ...),
3. accélérateurs de calcul (GPU ou autres) : Cuda, OpenCL,
4. outils PGAS ("partitionned global addressing system") : pour information (faibles performances),

on testera plusieurs combinaisons.

Machines multi-nœuds et multi-cœurs

Pour utiliser la puissance de calcul de ce type de machine, on a souvent le choix entre 2 possibilités :

- **MPI** pour le parallélisme multi-nœuds et **MPI** pour le parallélisme multi-cœurs.
- **MPI** pour le parallélisme multi-nœuds et **OpenMP** pour le parallélisme multi-cœurs.

Sur des simulations de taille petite ou moyenne, on préfère souvent le premier choix pour des raisons de simplicité.
Par contre, sur des simulations de (très) grande taille, le “tout MPI” atteint plus rapidement ses limites d’utilisation.

Avantages/inconvénients du tout MPI:

- ⊕ programmation MPI nœuds-cœurs identique à la programmation MPI entre les nœuds
- ⊖ le nombre de processus MPI augmente plus vite (n° cœurs $\times$ n° nœuds), les structures internes de MPI, les tampons utilisés pour les communication prennent plus de place(on atteint les limites sur des simulations très importantes)
- ⊖ le temps consacré aux communications MPI augmente (et donc les speedups diminuent)

Avantages/inconvénients de la combinaison MPI-OpenMP:

- ⊕ les possibilités en nombre de nœuds-cœurs sont plus importantes
- ⊖ la programmation MPI sur les nœuds et OpenMP sur les cœurs est plus complexe
- ⊖ l'amélioration des performances n'est pas toujours évidente