

Consignes exercices MPI-fortran

Je vais essayer de te faire appréhender au mieux ce qu'est le parallélisme MPI à travers quelques exercices simples.

Exo 1 Ce premier exo est simple. On va :

- 1) télécharger un compilateur fortran (gfortran)
- 2) télécharger les outils pour utiliser mpi
- 3) compiler notre premier code fortran/mpi
- 4) et l'exécuter

Pour ce faire :

- 1) dans le terminal taper : `sudo apt install gfortran` (si ne marche pas suivre les recommandations du terminal)
- 2) dans le terminal taper : `sudo apt install libopenmpi-dev`
- 3) pour compiler taper : `mpif90 -o exo1 exo1.f90` (nom du compilateur / niveau de compilation (ici 0) / nom de l'exécutable / fichier à compiler)
- 4) pour exécuter taper : `mpirun -n X exo1` (mot clé mpirun ou mpiexec / -n (ou -np) / nombre de processeur (remplacer X par 1,2...42..Etc) / nom de l'exécutable)

Question : Que se passe t'il quand tu changes le nombre de processeur ?

Quand on lance le programme avec 1 seul processeur on dit que le programme est lancé en "séquentiel". Tous les travaux que tu as pu faire jusqu'ici (fortran / C++) on toujours été lancé en séquentiel. Lancer le code sur plusieurs processeurs permet de gagner du temps en décomposant au mieux le travail. Un code parallèle performant est un code ou le travail est réparti au mieux.

Il faut comprendre qu'un programme finira son execution quand tout les processeurs finiront leurs parts du travail. Il est donc inutile d'avoir un processeur qui travaille plus que les autres. Il faut éviter cela au maximum.

Bien entendu ce code est stérile du point de vue de la programmation car ici tous les processeurs disent "Bonjour". On pourrait envisager, dans l'éventualité où le programme doit effectivement dire "Bonjour" de demander à un seul processeur de faire ce travail. Ceci sera ce que l'on verra dans l'exo2.

Exo 2 Premières fonctions MPI

L'objectif est d'améliorer le "bonjour" de l'exercice 1. On va chercher à obtenir un programme qui écrit "Bonjour je me présente, je suis le processeur numero : X sur Xtotal processeurs".

Pour ce faire on va utiliser 2 nouvelles fonctions MPI : `MPI_COMM_RANK` et `MPI_COMM_SIZE`. Le premier permet d'obtenir quel est le numero du processeur courant (rank) et le second le nombre total de processeur utilisé (size) ce nombre total est la valeur X choisie lors de l'exécution (cf exo1).

Ton exercice se résume à compléter le fichier `exo2.f90` jusqu'à ce que le programme compilé donne le message attendu. Pour le compléter tu auras besoin du fichier Aide mémoire qui te sera utile toute ta vie (du moins tant que tu codes du OPEN-MPI/Fortran).

Tu as réussi ? Super ! Résumons ce que tu as appris durant cette exercice 2.

- Tu as appris à utiliser le document pdf d'aide mémoire
- Tu as appris 2 nouvelles fonctions MPI (`MPI_COMM_SIZE` et `MPI_COMM_RANK`)
- Tu as remarqué que la première entrée de ces fonctions était déjà remplie avec la variable `MPI_COMM_WORLD`. Cette variable est une constante intrinsèque au MPI que l'on nomme communicateur. Elle permet de créer une structure de communication entre les processeurs. Par défaut c'est toujours le communicateur de base qui est utilisé (`MPI_COMM_WORLD`). Il faut savoir que l'on peut envisager de personnaliser son propre communicateur. Je ne vais pas t'apprendre à faire ça car cela relève des premiers cours de MPI en 3ème année.

Dans le prochain exercice on va faire travailler un peu chacun de nos processeurs et l'on va leur demander de communiquer leurs résultats à chacun des autres processeurs.