Consignes exercices MPI-fortran

Je vais essayer de te faire appréhender au mieux ce qu'est le parallèlisme 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'executer

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'executable / fichier à compiler)
- 4) pour executer 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'executable)

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 ecrit "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'execussion (cf exo1).

Ton exercice se résume à completer le fichier exo2.f90 jusqu'à ce que le programme compilé donne le message attendu. Pour le completer 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 le première entrée de ces fonctions était éjà 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 struture 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 ca car cela relève des premiers cours de MPI en 3eme année.

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

Exo 3 Paralléliser une somme

Dans cet exercice on va partager au mieux le calcul de PI réalisé via la formule de Leibniz

$$\Pi = 4 \sum_{k=0}^{\infty} \frac{(-1)^i}{(2i+1)} \tag{1}$$

Nous allons donc chercher à calculer la sommes des 10^9 premiers termes. En séquentiel, un seul processeur calculerait et sommerait 10^9 valeurs. Ici chaque processeur va calculer dans son coin sa partie de la somme. Et grâce à la commande MPI_ALLREDUCE nous allons sommer tous les résultats locaux (connu par un seul processeur) dans une variable commune a toute, une variable dite globale comme le montre le schéma suivant :

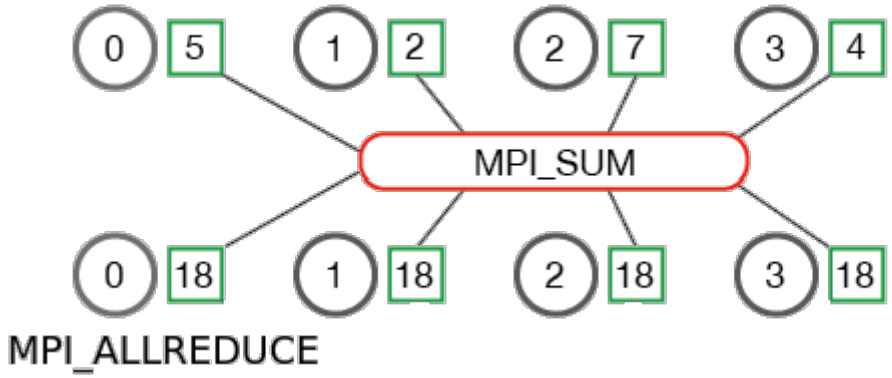

Figure 1: Schéma MPI_ALLREDUCE

Voilà comment sommer le résultat de chaque processeur. Mais préalablement il faut calculer correctement (et judicieusement) le resultat local sur chaque processeur. Pour se faire chaque processeur va devoir faire la somme entre un indice *i_begin* et *i_end* comme le montre l'image suivante :

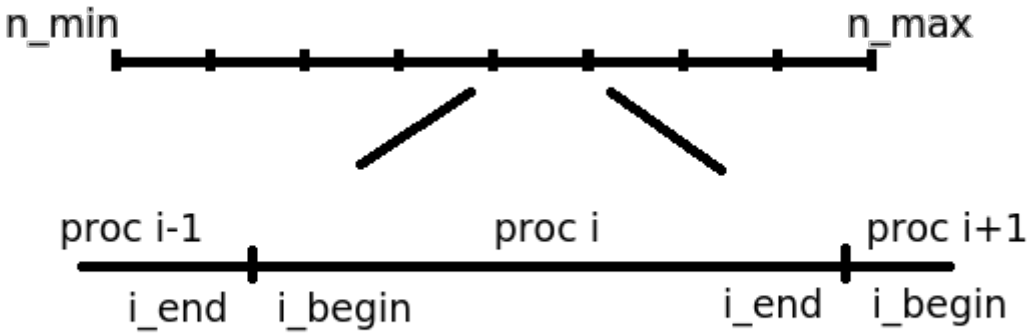

Figure 2: Décomposition de la charge de calcul par processeur

L'objectif de cet exercice est donc dans un premier temps de remplir les fonctions MPI du fichier exo3.f90 puis de remplir la routine fortran nommée *charge* qui devra calculer correctement pour chaque processeur les valeurs i_begin et i_end.

L'exercice sera bien réalisé si tu obtiens une approximation cohérente de Π pour différent nombre de processeur alloué au calcul.

Tu pourras ensuite t'interesser aux termes "Efficacité" ou "Speed-up" d'un programme en regardant leurs définitions et leurs formules sur internet.

Dans un dernier temps tu devras tracer ces courbes pour nb_proc allant de 1 à 16.

Une fois ceci fait, selon toi, combien de processeur possède ton ordinateur ?

Tips : Précède ta commande d'execution par la mot clé *time* pour avoir une estimation du temps de calcul du problème.