TP de programmation parallèle avec MPI

Université Lorraine - 2022 - 2023

Démarrage

On rappelle que la bibliothèque `MPI` délivre un ensemble de fonctions de communication entre processus lourds. Différentes distributions de `MPI` existent pour différents langages tels que `C/C++`, `Fortran` ou `Python`. Dans ce TP, nous allons utiliser la version `C/C++` de `MPI` via la distribution `Open MPI` (voir le document d'installation sur Arche). La compilation `MPI` nécessite un *meta*-compilateur, généralement nommé `mpicc` et `mpic++`. Ce *meta*-compilateur utilise un compilateur standard en amont (`gcc` par exemple).

La commande de compilation a la forme générale suivante :

```
mpicc source.c -o executable
```

Un fichier `Makefile` de compilation automatique est fourni avec les fichiers sources et il vous suffira donc d'utiliser la commande `make` pour compiler.

Pour exécuter un programme `MPI`, il faut démarrer les processus sur les différentes machines concernées. Pour cela, il faut utiliser la commande spécifique `mpirun` qui est fournie dans l'environnement d'exécution `MPI`. La commande `mpirun` a plusieurs options possibles dont les deux principales sont :

- `-n <int>` : qui indique le nombre (entier) de processus à lancer.
- `-machinefile <fichier>` : qui donne la liste des machines (dans un fichier texte) sur lesquelles lancer les processus. Il faut bien sûr que les connexions à ces machines soient possibles, sans interaction (mot de passe), en configurant `ssh` par exemple.

Il est également possible de lancer les processus sur une même machine. C'est ce que nous allons utiliser dans ce TP. Dans ce cas, le fichier des machines n'est pas nécessaire et on obtient la commande suivante pour lancer 8 processus sur la machine locale :

```
mpirun -n 8 -host localhost:8 ./executable
```

Une archive est disponible sur le site Arche du cours, qui contient les fichiers sources correspondant aux différents exercices de ce TP.

Exercice 1 : Démarrage du système de communication

Le premier programme, contenu dans le fichier source `creation.c`, démarre le système de communication `MPI` et effectue quelques affichages pour chaque processus.

a) Exécutez le programme avec 1, 2, 4 et 8 processus. Que constatez-vous ?

b) Exécutez le programme directement, sans utiliser la commande `mpirun`.
Que pouvez-vous en déduire ?

c) Exécutez le programme plusieurs fois avec 4 processus. Que constatez-vous concernant l'ordre des affichages ?

d) Ajoutez des communications entre les processus pour ordonner les affichages selon les numéros des processus.

Exercice 2 : Calcul de π

Le programme suivant, contenu dans le fichier source `pi.c`, effectue le calcul de π avec un seul processus.

a) Exécutez le programme avec plusieurs quantités de trapèzes (option `-n` suivie d'un nombre entier). Observez l'erreur sur π et notez les temps de calcul.

b) Modifiez le programme afin de répartir les trapèzes sur l'ensemble des processus. Les intervalles `[deb,fin]` des processus doivent former une partition de l'intervalle `[0,1]`. Exécutez le programme avec plusieurs processus. Que constatez-vous concernant le temps de calcul ? Et le résultat ?

c) Ajoutez les communications nécessaires pour fusionner tous les résultats partiels sur le processus 0.

Exercice 3 : Comptage des occurrences des lettres dans un texte

Le programme contenu dans le fichier source `occurrences.c`, génère un texte aléatoire de taille donnée (option `-t`) et compte les occurrences des lettres de l'alphabet sans distinction des minuscules et majuscules. La graine aléatoire peut être spécifiée via l'option `-g`. Dans ce programme, une version séquentielle et une version parallèle `MPI` sont prévues mais seule la version séquentielle est opérationnelle.

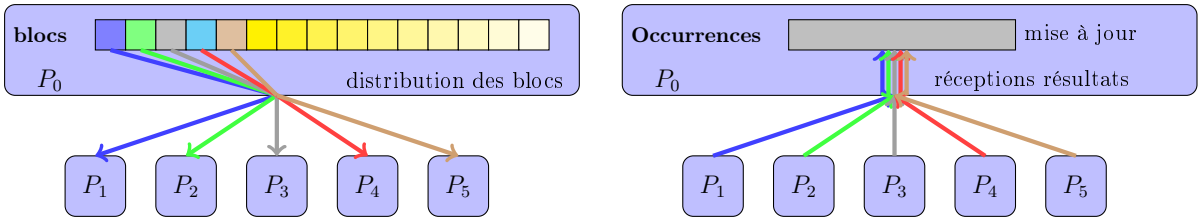
a) Exécutez le programme avec plusieurs tailles de texte. Que constatez-vous sur les fréquences d'occurrences ?

Pour paralléliser ce calcul, nous utilisons le schéma *maître-travailleurs* dans la fonction `compte_occurrences_mpi()`. Le principe est qu'un processus particulier (le *maître*) distribue aux autres processus (les *travailleurs*) le texte initial par blocs de taille donnée (option `-b`).

Dans le processus maître, une première distribution est faite d'un bloc de texte à chaque travailleur. Puis, le maître attend les résultats des travailleurs. À chaque réception d'un résultat, le maître met à jour le résultat final (tableau des occurrences) et renvoie au travailleur correspondant un autre bloc de texte, s'il en reste. Lorsqu'il n'y a plus de bloc de texte à envoyer, le maître envoie un message vide pour indiquer la fin du travail.

Du côté des travailleurs, on effectue un boucle dans laquelle on reçoit un bloc de texte provenant du maître et si celui-ci n'est pas vide (taille nulle), on le traite et on renvoie au maître le tableau des occurrences pour ce bloc. Si le message est vide, cela indique la fin du travail et la boucle s'arrête.

À la fin, le résultat final est disponible sur le processus maître. La figure ci-dessous illustre la stratégie globale avec six processus et P_0 comme maître.



b) Complétez les parties vides du schéma parallèle. Il peut être plus facile de commencer par les travailleurs (schéma simple de réception-travail-envoi), puis de compléter progressivement le maître.