TP N° 4
Programmation parallèle-MPI

Objectif :

- Utiliser effectivement les fonctions de **MPI** pour paralléliser un programme en vue d'une exécution sur une architecture parallèle à mémoire distribuée.

Outil : Compilateur **C** avec le package **OpenMPI**.

Dans ce TP, il est demandé d'implémenter une distribution de données sur un ensemble de processus selon le modèle maître-esclave, en utilisant les fonctions de communication point à point. Effectivement, il est demandé d'avoir un processus maître au niveau duquel les données à répartir sont stockées, et plusieurs processus esclaves. Le processus maître distribue à la demande des processus esclaves, des portions de données pour que ces derniers effectuent un travail (qui peut se faire en parallèle). Les résultats obtenus par les processus esclaves sont transmis au processus maître pour être stockés.

Ci-dessous, sont résumées les tâches des processus esclaves et celles du processus maître, ainsi que quelques contraintes à respecter afin d'implémenter le code.

- **Tâches d'un processus esclave (chaque processus esclave les exécute)**
  1. Envoi de la première demande de données (demande d'une portion de taille *chunksize*) au processus maître.
  2. Réception d'une portion de données de taille *chunksize*, de la part du processus maître.
  3. Traitement des données reçues (mise au carré des éléments de la portion reçue).
  4. Transmission des données traitées (envoi d'une portion de données de taille *chunksize*) au processus maître. Cette transmission de résultat est aussi considérée comme une nouvelle demande de données de taille *chunksize*.

5. Les étapes **3** et **4** sont répétées tant qu'il n'y a pas réception d'ordre de fin de la part du processus maître (l'étape **2** est toujours répétée, il y a donc soit réception d'une portion de données, soit réception d'un ordre de fin).
6. La réception de l'ordre de fin marque la fin des tâches des processus esclaves.

- **Tâches du processus maître**
  1. Réception d'une demande de données d'un esclave (si c'est autre que la première demande du processus esclave, alors il y a stockage des données transmises par celui-ci. La portion de données reçue est de taille *chunksize* et elle est récupérée dans une zone mémoire contiguë de taille *n*chunksize* du processus maître, selon l'ordre de réception).
  2. Envoi d'une portion de données de taille *chunksize* (s'il reste des portions non encore attribuées) au processus esclave demandeur (dont la requête est traitée à l'étape **1**). Autrement, envoi à tous les processus esclaves d'un ordre de fin et arrêt de traitement des demandes de données.
  3. Les étapes **1** et **2** sont répétées pour permettre au processus maître de traiter les demandes de tous les processus esclaves.

- **Contraintes à respecter**
  1. Le code doit permettre de traiter n'importe quel nombre de processus créés supérieur ou égal à 2.
  2. Le code doit permettre à n'importe quel processus d'être le processus maître pour différentes exécutions.
  3. Le code doit permettre de traiter différentes tailles pour le tableau de données à distribuer et pour *chunksize* (avec *chunksize* qui divise la taille du tableau et qui donne *n*).
  4. Le code ne doit contenir que des fonctions de communication point à point.