Structures de données et tri avec MPI

Résumé: Dans ce TD, nous mettrons en œuvre un tri en parallèle en MPI basé sur le réseau de tri pair/impair vu en cours. Le tri se fera sur une structure de données complexe, ce qui permettra de manipuler un peu les types MPI.

1 Quelques nouvelles fonctions MPI

Il est possible de créer des types MPI afin de faciliter la manipulation de structures complexes et de s'affranchir de conversions explicites de données lorsque la plateforme de calcul est hétérogène. Nous allons utiliser pour cela les fonctions suivantes :

▷ `MPI_Address` Sur certain systèmes, la valeur renvoyée par cette fonction sera la même que celle que vous auriez récupérée en utilisant l'opérateur `&` du C mais cela n'est pas obligatoire. Il vaut donc mieux utiliser les adresses renvoyées par cette fonction pour calculer les offset des différents champs d'une structure.

```
int MPI_Address( void *location, MPI_Aint *address)
```

▷ `MPI_Type_struct` Cette fonction permet de construire un type MPI structuré. Le prototypage est le suivant :

```
int MPI_Type_struct(
      int count, int blocklens[],
      MPI_Aint indices[], MPI_Datatype old_types[],
      MPI_Datatype *newtype )
```

- `count` représente le nombre de champs de la structure (c'est aussi le nombre d'éléments des différents tableaux passés en arguments)
- `blocklens[i]` représente le nombre d'éléments dans le champs `i`.
- `indices[i]` représente l'offset du champs `i` dans la structure C.
- `old_types[i]` représente le type des éléments du champs `i`.
- `newtype` est le type structuré créé par l'appel à `MPI_Type_struct`.

▷ `MPI_Type_commit` permet de rendre public un type MPI et doit être appelé avant toute utilisation du nouveau type.

Enfin, voici deux autres fonctions qui nous seront utiles pour distribuer et récupérer les données :

▷ `MPI_Scatter` Cette fonction permet de distribuer des données à un groupe de processeurs.

```
int MPI_Scatter (
      void *sendbuf, int sendcnt, MPI_Datatype sendtype,
      void *recvbuf, int recvcnt, MPI_Datatype recvtype,
      int root, MPI_Comm comm )
```

- `root` est l'indice du processeur qui va effectuer les émissions.
- `sendbuf` est l'adresse du buffer d'émission (pertinent uniquement sur le processeur `root`).
- `sendcnt` est le nombre d'éléments qui va être envoyé à chaque processeur (pertinent uniquement sur le processeur `root`).
- `sendtype` est le type MPI des objets du buffer (pertinent uniquement sur le processeur `root`).
- `recvbuf` est l'adresse du buffer de réception.
- `recvcnt` est le nombre d'éléments qui va être reçu par chaque processeur.
- `recvtype` est le type MPI des objets que les processeurs vont recevoir.

▷ `MPI_Gather` Cette fonction permet de rassembler vers un processeur donné des données distribuées sur un groupe de processeurs.

```
int MPI_Gather (
      void *sendbuf, int sendcnt, MPI_Datatype sendtype,
      void *recvbuf, int recvcnt, MPI_Datatype recvtype,
      int root, MPI_Comm comm )
```

- `root` est l'indice du processeur qui va récupérer les données.
- `sendbuf` est l'adresse du buffer d'émission.
- `sendcnt` est le nombre d'éléments qui va être envoyé à chaque processeur.

- `sendtype` est le type MPI des objets du buffer .
- `recvbuf` est l'adresse du buffer de réception (pertinent uniquement sur le processeur `root`).
- `recvcnt` est le nombre d'éléments qui va être reçu de la part de chacun des processeurs (pertinent uniquement sur le processeur `root`).
- `recvtype` est le type MPI des objets que le processeur `root` va recevoir (pertinent uniquement sur le processeur `root`).

Au cas où ces explications ne suffiraient pas, je vous invite à aller voir les sites suivants :
- http://www-unix.mcs.anl.gov/mpi/tutorial/gropp/talk.html
- http://www.netlib.org/utk/papers/mpi-book/mpi-book.html

Vous pouvez également aller lire les pages de man qui sont disponibles à peu près n'importe où.

2 Tri en parallèle

Vous trouverez dans http://www-id.imag.fr/Laboratoire/Membres/Legrand_Arnaud/algopar/MPI/src/ un canevas pour le programme de tri (`parsort_init.c`) ainsi qu'un fichier contenant des informations (désormait totalement obsolètes) sur les utilisateurs du réseau de l'école (`liste.txt`). Dans sa version initiale, ce programme lit ce fichier et stocke les valeurs dans un grand tableau sur le processeur 0. Il vous appartient donc de le modifier afin de distribuer ce tableau sur les différents processeurs, de faire un tri pair/impair puis de rassembler les résultats sur le processeur 0.

Une solution possible est disponible à la même adresse dans le fichier `parsort.c`.