INF7235 : Laboratoire #9
Produit parallèle distribué de matrices avec MPI et diffusion asynchrone des données

27 mars 2017

Introduction

L'objectif de ce troisième (et dernier) laboratoire MPI est de vous familiariser encore un peu avec les communicateurs — qui définissent des sous-groupes de processus — et avec l'envoi et réception de messages en mode `I`*mmediate*, plus spécifiquement, avec `Ibcast` — mode qui permet aux communications de s'effectuer **en même temps que certains calculs sont effectués**.

Pour obtenir le code source

```
$ git clone http://www.labunix.uqam.ca/~tremblay/git/MultMat.git
```

Ce que vous devez faire

Le programme `mm-bcast.c` met en oeuvre la solution de Mattson, Sanders et Massingil vue en cours pour le produit de matrices distribué. Comme dans le code vu dans les diapositives, la communication des blocs se fait à l'aide d'une procédure auxiliaire `bcastBloc`, laquelle utilise l'opération `MPI_Bcast` pour diffuser un bloc aux processus appropriés.

Un fichier `mm-ibcast.c` vous est fourni. Pour l'instant, le code est identique à celui de `mm-bcast.c`, mais avec deux procédures additionnelles que vous aurez besoin d'utiliser : `ibcastBloc` et `swap`.

Dans le fichier `mm-ibcast.c`, vous devez modifier le code du programme principal **pour que les calculs et les communications se fassent de façon concurrente** — donc en **initiant** les requêtes pour le prochain bloc **pendant que** les calculs sur le bloc courant sont effectués!

Pour ce faire, il vous faut utiliser la procédure additionnelle `ibcastBloc`. Notez qu'il vous faut aussi utiliser la procédure `bcastBloc` telle quelle, mais uniquement pour le premier bloc.

La cible par défaut du `makefile` — commande «`make`» sans argument — compile le fichier `mm-ibcast.c`. Lorsque le fichier compile correctement, il suffit d'exécuter «`make run`» pour lancer son exécution — ou de modifier la cible par défaut dans le `makefile`.

Informations additionnelles

Voici la description de la méthode `MPI_Ibcast` :

```
int MPI_Ibcast(void *buffer, int count, MPI_Datatype datatype,
               int root, MPI_Comm comm, MPI_Request *request)

Broadcasts a message from the process with rank root to all
other processes of the group

Input/Output Parameters
buffer   starting address of buffer (choice)
count    number of entries in buffer (integer)
datatype data type of buffer (handle)
root     rank of broadcast root (integer)
comm     communicator (handle)

Output Parameters
request Request (handle, non-blocking only).
```

Voici les grandes lignes du programme modifié qui effectue les calculs et les communications de façon concurrente :

```
DEBUT
  On effectue le transfert des données pour la phase 0 (bcastBloc)

  POUR chacune des NB phases de calcul FAIRE
    On amorce le transfert des données pour la prochaine phase (ibcastBloc)

    On traite les blocs de données déjà disponibles



    On attend que les données pour la prochaine phase soient transmises/arrivées
  FIN

  On traite le dernier bloc de donnees
FIN
```

Note : Pour que les calculs et les communications puissent se faire de façon concurrente, vous aurez besoin de tampons (*buffers*) auxiliaires... qu'il faudra interchanger avant de débuter une nouvelle itération — d'où la procédure `swap`.