Introduction à MPI

Il s'agit de s'initier à MPI et d'observer les comportements de la latence réseau.
Pour éviter de vous marcher trop les uns sur les autres, les expériences seront d'abord menées entre votre propre machine et celle de votre voisin : **modifiez** le fichier `mymachines` pour y mettre le nom de la machine de votre voisin.
Note : il peut arriver que l'exécution échoue :

```
mpirun.openmpi was unable to launch the specified application as it could not access
or execute an executable:

Executable: ./ping
```

C'est simplement parce que le fichier n'a pas eu le temps d'apparaître, via le réseau, sur l'autre machine. Relancez la commande, et cette fois cela fonctionnera.

1 Avant toutes choses

Pour pouvoir éviter de taper son mot de passe, si vous ne l'avez pas fait lors d'un TP précédent, générez une paire clé publique/privée :
`ssh-keygen -t dsa`
Validez autant de fois qu'il le faut (gardez les valeurs par défaut). Utilisez une passphrase vide, à moins que vous sachiez gérer un agent ssh. Enfin, autorisez l'utilisation de la clé :
`cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys`
Vérifiez avec un `ssh localhost` que vous n'avez pas à taper de mot de passe pour vous connecter à la main à la machine de votre voisin. Il faudra au besoin confirmer l'identité de la machine.
Il se peut que ssh vous indique `WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!` C'est en général simplement parceque les machines ont été réinstallée et ont donc changé d'identité. Il faut supprimer l'ancienne identité (*Offending key*) du fichier `.ssh/known_hosts` à la ligne indiquée.

2 Découverte

Le programme `ping.c` est un exemple typique de programme MPI.
Pour lancer ce programme MPI, un simple `make run` devrait suffire.
C'est le même programme qui est lancé sur différentes machines (appelées *nœuds* et numérotés à partir de 0), et l'on utilise quelques fonctions pour communiquer.

- `MPI_Init(&argc,&argv);` permet d'initialiser la bibliothèque MPI : on lui passe l'adresse de `argc` et `argv` pour qu'elle puisse savoir comment les nœuds doivent se connecter.
- `MPI_Comm_size(MPI_COMM_WORLD,&num);` permet de récupérer le nombre de nœuds lancés.
- `MPI_Comm_rank(MPI_COMM_WORLD,&myid);` permet de récupérer son propre numéro de nœud (entre 0 et `num`−1).
- `MPI_Send(buf,N,MPI_BYTE,1,0,MPI_COMM_WORLD);` envoie un tableau `buf` de `N` octets (`MPI_CHAR`) au nœud 1, avec le tag 0.
- `MPI_Recv(buf,N,MPI_CHAR,0,0,MPI_COMM_WORLD,&status);` réceptionne un tableau `buf` de `N` caractères (`MPI_CHAR`) envoyé par le nœud 0 avec le tag 0 (il faut donc que ce soit le même que du côté émetteur). L'état de la réception est stocké dans la variable `status`.
- `MPI_COMM_WORLD` est le communicateur utilisé : il inclut tous les nœuds lancés.

On pourra bien sûr garder sous les yeux les pages de man de ces fonctions MPI, mais sur `http://mpi.deino.net/mpi/mpi_functions/` la documentation est plus fournie et d'autres exemples de codes sont disponibles. Une documentation très détaillée est disponible sur `http://www.netlib.org/utk/papers/mpi-book/mpi-book.html` .

3 Latence

Pour l'instant, ce programme ne fait qu'envoyer un caractère d'une machine à une autre ("ping") et de mesurer le temps pris par les fonctions d'envoi et de réception.
Est-ce une manière correcte de mesurer le temps que prend la communication ?
Complétez le programme pour que la deuxième machine renvoie ce caractère ("pong") et la première machine le réceptionne, et mesurez la latence de l'aller-retour. Lancez plusieurs fois, constatez que la mesure fluctue. Doit-on prendre la valeur minimum, la moyenne, autre chose ?
Pour effectuer ce calcul automatiquement, utilisez une boucle effectuant ce jeu de ping-pong 300 fois. Ajoutez avant celle-ci une première boucle, non mesurée, effectuant la même chose, mais quelques dizaines de fois pour « préchauffer les fils » (c'est-à-dire, passer dans le code de MPI quelques fois pour que les heuristiques des caches et prédiction de branchement, etc. se stabilisent).
Ajoutez une boucle pour faire progresser `N` de manière géométrique (de facteur 2) jusqu'à $1024 * 1024$. Utilisez `gnuplot` pour tracer une courbe en fonction de la taille. Utilisez la commande `set logscale` pour passer en échelle logarithmique.

4 Isend

Dans le code du nœud 0, remplacez `MPI_Send()` par le couple « `MPI_Isend()` puis `MPI_Wait()` » : quasiment rien n'est changé, on donne juste à `MPI_Isend` l'adresse d'un tampon de requête de type `MPI_Request`, que l'on fournit ensuite à `MPI_Wait` pour attendre la fin de la requête d'émission. Constatez que cela ne change pas la latence.
Changez les `gettimeofday()` pour mesurer séparément le temps mis par `MPI_Isend()` et par `MPI_Wait` (le plus simple est de cumuler les différences). Tracez une jolie courbe : dans `data`, rentrez les données ainsi :

```
1 temps_MPI_Isend_1 temps_MPI_Wait_1
2 temps_MPI_Isend_2 temps_MPI_Wait_2
...
```

Et tapez ceci dans `gnuplot` :

```
set logscale
plot "data" using ($1):($2) with linespoints, "data" using ($1):($3) with linespoints, \
"data" using ($1):($2)+($3) with linespoints
```

pour tracer les courbes et leur somme en même temps.
On aperçoit vraiment nettement une cassure, qui correspond au changement de stratégie entre envoi direct et envoi par rendez-vous : avec rendez-vous, ce n'est alors plus `MPI_Isend()` qui fait l'envoi effectif des données, mais `MPI_Wait()`.

5 Et la bande passante ?

Comment mesurer la bande passante (en méga-octets échangés par seconde) ? Tracez de même une courbe.

6 Un anneau

Généralisez le programme à n machines : le nœud 0 envoie les données au nœud 1, qui le retransmet au nœud 2, etc jusqu'au nœud $n-1$ qui l'envoie de nouveau au nœud 0 (modifiez l'option `-np` dans `Makefile` pour exécuter plus que 2 processus, il faudra ajouter d'autres noms de machines dans `mymachines`). Comment la latence croît-elle avec n ?

7 Mémoire partagée vs. réseau

En mettant dans le fichier `mymachines` plusieurs fois le même nom de machine, les processus seront lancés sur les différents processeurs de cette machine et les communications se feront par mémoire partagée. Puisque vos machines ont 8 cœurs, répartissez 8 processus (en mettant `-np` à 8 dans le fichier `Makefile`). Désactivez la prise de verrou, car sinon cela ne pourrait pas se lancer puisqu'il essaierait de prendre le verrou plusieurs fois. Observez la latence obtenue par le programme de test en anneau.