TRAVAUX PRATIQUES DE PARALLELISME
2010 – 2011
MPI, Communication par messages

Objectif pédagogique :

Premier contact avec MPI et écriture d'une application simple de calcul parallèle

Nb : pour des raisons de sécurité, l'environnement est pour l'instant restreint à l'exécution d'un programme MPI sur une seule machine.

Mise en place de l'environnement :

Attention, il y a plusieurs versions de MPI installées à l'INSA. **Pour ce TP**, vous utiliserez la version "mpich2" dont les exécutables se trouvent sous
`/home-reseau/pazat/bin`
Créez d'abord un fichier `.machines` sous votre home dir. Ce fichier doit contenir une liste de machines qui seront utilisées pour exécuter vos programmes MPI (1 nom de machine par ligne)
Pour compiler un programme:
`/home-reseau/pazat/mpicc` <nom du programme>
Pour exécuter un programme:
`/home-reseau/pazat/mpiexec -n` <nb processus> <nom du programme>

test: copiez, compilez et exécutez le programme `monprog.c`
Vous devez obtenir

```
Hello world from node 0 of 1 nodes
Fini
```

```
/home-reseau/pazat/mpiexec  -n 3 monprog
```

vous devez obtenir (pas forcément dans cet ordre)

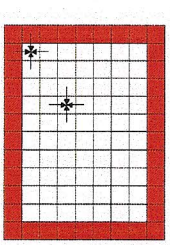
```
Hello world from node 0 of 3 nodes
Hello world from node 2 of 3 nodes
Fini
Hello world from node 1 of 3 nodes
Fini
Fini
```

Premier exemple

Copiez, compilez et exécutez le programme `teste.c` avec un nombre variable de processus. Ce petit exemple contient l'essentiel de ce qu'il faut savoir pour écrire des programmes MPI simples.

L'application

L'application à réaliser simule la propagation de la chaleur sur une plaque rectangulaire dont les bords sont en contact avec un milieu uniformément chaud. La plaque est « discrétisée » sous forme d'un matrice de points. Une matrice représente la température de la plaque en chacun de ses points. On va calculer de manière itérative la répartition de la chaleur sur cette plaque grâce à l'équation suivante :



$$T_{t+1}(i,j) = (T_t(i,j+1)+T_t(i,j-1)+T_t(i+1,j)+T_t(i-1,j)+T_t(i,j))/5$$

Le calcul se termine lorque l'écart moyen est inférieur à un certain seuil :

$$\Delta = \sum_{i,j} |T_{t+1}(i,j) - T_t(i,j)|$$

Pour simplifier l'écriture de l'algorithme, les bords de la matrice représentent le milieu extérieur. Il n'y a ainsi pas de cas particulier à traiter.

L'algorithme a donc la forme suivante :

```
double T[N+2][M+2], T1[N+2][M+2]
```

```
Initialiser (T=0 sauf les bords = MAX) ;
Faire
  delta=0;
   Pourtout i=1..N,j=1..M :
    T1[i][j] = (T[i][j+1]+T[i][j-1]+T[i+1][j]+T[i-1][j]+T[i)[j])/5;
    delta = delta + |Tt+1(i,j)-Tt(i,j)| ;
    T = T1 ;
Jqa ( delta < seuil )
```

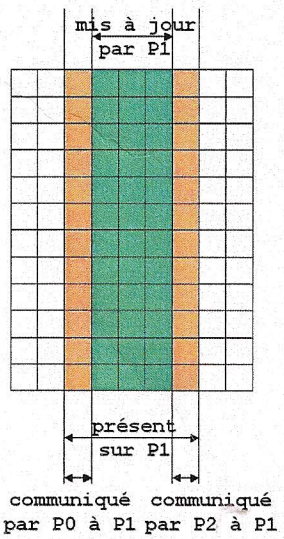
Attention : sur les bord (i=0 ou j=0 ou i=N+1 ou j=M+1) la valeur reste inchangée : il s'agit du milieu extérieur.

Question 1

Ecrire le programme séquentiel en C et le tester. On fera afficher le contenu de la matrice toute les K itérations.

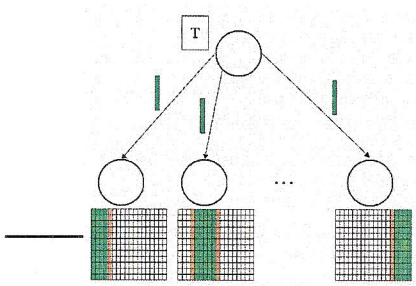
Question 2

Donner une mise en œuvre en MPI dans laquelle la matrice est distribuée en B blocs dans une seule dimension.



Le programme aura une forme « SPMD » (même code sur chaque processeur mais travaillant sur des données différentes. En plus de ces processus on aura un processus initiateur qui lancera le calcul et récupèrera un résultat (la copie de la matrice) à intervalles réguliers (toutes les K itérations). Pour gérer la terminaison, on pourra se contenter d'un algorithme centralisé : on transmet delta avec la matrice et si l'initiateur s'aperçoit que delta est inférieur au seuil il arrête (kill) les processus SPMD.

Recouvrement : du fait que l'on a besoin des 2 voisins pour réaliser un calcul, chaque processeur calculera une tranche de la matrice mais possèdera 2 colonnes de plus qui seront mises à jour à chaque itération par ses voisins . On recopiera donc à chaque itération une colonne de la matrice de Pi-1 vers Pi et de Pi+1 vers Pi.



Remarque

ce document ainsi que les programmes `monprog.c` et `teste.c` sont sous le répertoire « commun »

`/home-info/commun/4info/MPP/TP/tp3`