UPMC
SORBONNE UNIVERSITÉS

TME 3 - Minimum sur un arbre et Echo avec MPI

Exercice(s)

Exercice 1 – Calcul de minimum sur un arbre

Le but de cet exercice est d'utiliser MPI pour mettre en œuvre un calcul de minimum réparti. On suppose que la topologie sur laquelle ce calcul est effectué est une topologie en arbre, où toutes les liaisons sont bidirectionnelles. Comme les machines des salles de TME sont organisées en grappes, on est obligé de simuler cette topologie. Pour cela, on décide qu'un processus particulier, le processus 0, sera chargé d'initialiser l'application répartie. La structure du programme exécuté par chaque processus est donc la suivante :

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char* argv[]) {
   int nb_proc,rang;
   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &nb_proc);

   if (nb_proc != NB_SITE+1) {
      printf("Nombre de processus incorrect !\n");
      MPI_Finalize();
      exit(2);
   }

   MPI_Comm_rank(MPI_COMM_WORLD, &rang);

   if (rang == 0) {
      simulateur();
   } else {
      calcul_min(rang);
   }

   MPI_Finalize();
   return 0;
}
```

où `NB_SITE` est une constante donnant le nombre de nœuds de l'arbre. On a donc au total, en comptant le processus 0, `NB_SITE+1` processus. Le processus 0 transmet à chaque noeud participant au calcul de minimum les informations suivantes :

— son nombre de voisins,
— la liste de ses voisins,

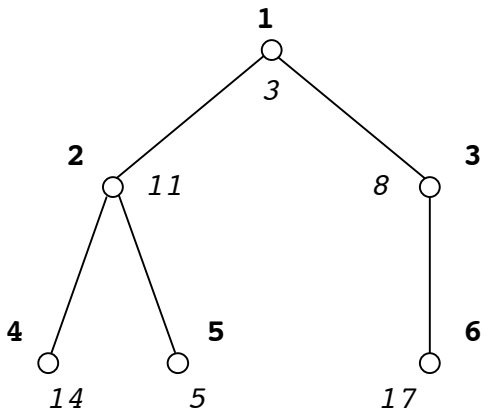

FIGURE 1 – Topologie en arbre

— la valeur de son minimum local.

Par exemple, on souhaite simuler l'arbre présenté dans la figure 1 où la valeur du minimum local de chaque site est indiquée en italique.

La fonction simulateur exécutée par le processus 0 est donnée par :

```
#define TAGINIT    0
#define NB_SITE 6

void simulateur(void) {
   int i;

   /* nb_voisins[i] est le nombre de voisins du site i */
   int nb_voisins[NB_SITE+1] = {-1, 2, 3, 2, 1, 1, 1};
   int min_local[NB_SITE+1] = {-1, 3, 11, 8, 14, 5, 17};

   /* liste des voisins */
   int voisins[NB_SITE+1][3] = {{-1, -1, -1},
         {2, 3, -1}, {1, 4, 5},
         {1, 6, -1}, {2, -1, -1},
         {2, -1, -1}, {3, -1, -1}};

   for(i=1; i<=NB_SITE; i++){
      MPI_Send(&nb_voisins[i], 1, MPI_INT, i, TAGINIT, MPI_COMM_WORLD);
      MPI_Send(voisins[i],nb_voisins[i], MPI_INT, i, TAGINIT, MPI_COMM_WORLD);
      MPI_Send(&min_local[i], 1, MPI_INT, i, TAGINIT, MPI_COMM_WORLD);
   }
}
```

La valeur `-1` est attribuée aux informations non significatives. `nb_voisins[i]` et `min_local[i]` donnent respectivement le nombre de voisins du site i et la valeur locale utilisée pour le calcul de minimum. `voisins[i][j]` est le (j+1)ème voisin du site i. MPI garantit le séquencement des messages entre deux sites, par conséquent les informations envoyées par le processus 0 seront reçues dans le même ordre sur les sites distants.

On se propose d'écrire la fonction `calcul_min(rang)` réalisant un calcul de minimum réparti en appliquant un algorithme de parcours d'arbre. On rappelle le principe de cet algorithme :

$\mathbf{R_p}$: { *un message* $\langle x \rangle$ *arrive de q* }
recevoir $\langle x \rangle$;
$Recu_p$[q] = vrai;

$\mathbf{S_p}$: { $Sent_p$ = *faux et* $\exists q \in Vois_p$, $\forall r \in Vois_p \backslash \{q\}$: $Recu_p[r]$ }
envoyer $\langle x \rangle$ à q;
$Sent_p$ = vrai;

$\mathbf{D_p}$: { $\forall q \in Vois_p$: $Recu_p[q]$ }
décision

La version que l'on veut implémenter est celle du calcul de minimum avec annonce, vue en TD : les processus parvenus à une décision informent ceux de leurs voisins qui ne connaissent pas encore le résultat.

Les fonctions `main` et `simulateur` sont disponibles sur la page web de l'UE.

Question 1
Implémentez sous MPI la fonction `calcul_min(rang)` qui réalise un calcul de minimum avec annonce, en affichant l'identité des processus décideurs et la valeur du minimum calculé.

Question 2
Testez votre implémentation sur la topologie ci-dessus, en exécutant tous les processus sur des machines différentes. Répétez l'exécution plusieurs fois. Que constatez-vous ?

Exercice 2 – L'algorithme ECHO
Le but de cet exercice est d'utiliser MPI pour mettre en œuvre un calcul de minimum réparti sur une topologie quelconque où toutes les liaisons sont bidirectionnelles. Comme pour l'exercice précédent, il est nécessaire de simuler cette topologie. Le programme principal est identique à celui de l'exercice précédent. La fonction simulateur correspondant à la topologie de la figure 2 est donnée par :

```
#define TAGINIT    0
#define NB_SITE 6

void simulateur(void) {
   int i;

   /* nb_voisins[i] est le nombre de voisins du site i */
   int nb_voisins[NB_SITE+1] = {-1, 3, 3, 2, 3, 5, 2};
   int min_local[NB_SITE+1] = {-1, 12, 11, 8, 14, 5, 17};

   /* liste des voisins */
   int voisins[NB_SITE+1][5] = {{-1, -1, -1, -1, -1},
            {2, 5, 3, -1, -1}, {4, 1, 5, -1, -1},
            {1, 5, -1, -1, -1}, {6, 2, 5, -1, -1},
            {1, 2, 6, 4, 3}, {4, 5, -1, -1, -1}};

   for(i=1; i<=NB_SITE; i++){
      MPI_Send(&nb_voisins[i], 1, MPI_INT, i, TAGINIT, MPI_COMM_WORLD);
      MPI_Send(voisins[i], nb_voisins[i], MPI_INT, i, TAGINIT, MPI_COMM_WORLD);
      MPI_Send(&min_local[i], 1, MPI_INT, i, TAGINIT, MPI_COMM_WORLD);
   }
}
```

On veut effectuer le calcul au moyen de l'algorithme ECHO dont on rappelle le principe :

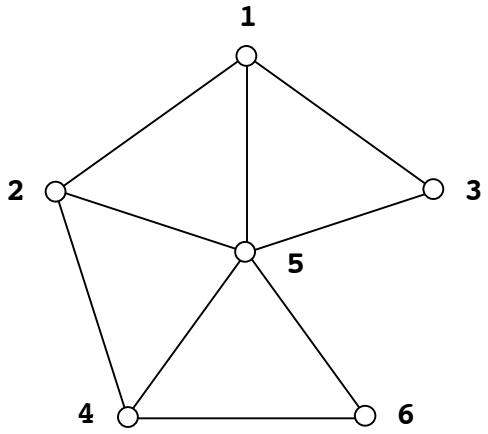

FIGURE 2 – Topologie pour l'exécution de ECHO

	p initiateur	p non initiateur
$\mathbf{S_p}$:	{ *spontanément, une fois* } $\forall q \in$ Vois$_p$ faire envoyer $\langle x \rangle$ à q;	{ $\forall q \in$ *Vois$_p$: Recu$_p$[q]* } envoyer $\langle x \rangle$ à pere$_p$;

$\mathbf{R_p}$:	{ *un message* $\langle x \rangle$ *arrive de q* }
	recevoir $\langle x \rangle$;
	Recu$_p$[q] = vrai;
		{ *un message* $\langle x \rangle$ *arrive de q* }
		recevoir $\langle x \rangle$;
		Recu$_p$[q] = vrai;
		si (pere$_p$ == nil) {
		pere$_p$ = q
		$\forall$ r $\in$ Vois$_p \backslash$ {q} faire
		envoyer $\langle x \rangle$ à r
		}

$\mathbf{D_p}$:	{ $\forall q \in$ *Vois$_p$: Recu$_p$[q]* } décision	

Les fonctions `main` et `simulateur` sont disponibles sur la page web de l'UE.

Question 1

Implémentez sous MPI la fonction `calcul_min(rang)` qui réalise un calcul de minimum réparti en appliquant l'algorithme ECHO.

Question 2

Appliquez votre programme sur la topologie fournie en exemple, en exécutant tous les processus sur des machines différentes.

Question 3

Modifiez votre implémentation pour que le calcul de minimum s'accompagne de la construction d'un arbre de recouvrement. Utilisez cet arbre pour réaliser une annonce du résultat du calcul de minimum.