Chapitre 6 : MPI

6.1 Une tâche MPI s'exécute sur deux noeuds. L'idée est d'échanger de l'information entre le noeud 0 et le noeud 1. Cependant, les tâches bloquent et donc ne se terminent jamais. Quel est le problème?

```
switch (rank) {
  case 0:
    MPI_Send(msg0, MSG_SIZE, MPI_INT, 1, 1, MPI_COMM_WORLD);
    MPI_Recv(msgr, MSG_SIZE, MPI_INT, 1, 1, MPI_COMM_WORLD,&status);
    break;
  case 1:
    MPI_Send(msg1, MSG_SIZE, MPI_INT, 0, 1, MPI_COMM_WORLD);
    MPI_Recv(msgr, MSG_SIZE, MPI_INT, 0, 1, MPI_COMM_WORLD,&status);
    break;
  default:
    break;
}
```

6.2 Un programme MPI est réparti sur un grand nombre de noeuds, chaque noeud contenant un vecteur qui donne la chaleur pour chaque point en X, le long d'une ligne horizontale qui correspond à un Y donné. Pour simuler la diffusion de chaleur dans une plaque, chaque noeud doit obtenir le vecteur de ses deux voisins, celui avec Y juste en-dessous et celui avec Y juste au-dessus. Chaque noeud a une position, rank, qui va de 0 à size - 1 et qui correspond à la coordonnée en Y. La fonction `itère_chaleur` reçoit en argument rp (rangée précédente), le vecteur de chaleur du noeud de rang inférieur (ou NULL pour la première rangée), rs (rangée suivante), le vecteur de chaleur du noeud de rang supérieur (ou NULL pour la dernière rangée), et rc (rangée courante), le vecteur de chaleur du noeud courant. Cette fonction modifie l'argument rc en y plaçant les nouvelles valeurs de chaleur. La boucle de calcul de cette application est fournie. Complétez cette boucle avec les énoncés (et commentaires appropriés) permettant d'effectuer efficacement ce travail.

```
int rank, size;
MPI_status status;
float rp[2048], rs[2048], rc[2048];
...
for(t = 0 ; t < max_time; t++) {
  /* Obtenir les valeurs des rangées précédentes et suivantes
     et leur fournir les nôtres. Nous sommes la rangée (noeud) rank,
     comprise entre 0 et (size - 1) et size est pair et > 1 */

  /* complétez cette section*/
  ...
```

```
    itere_chaleur(rp, rs, rc)
    ...
}
```

6.3 Chaque tâche MPI contient une matrice diagonale (tous les éléments sur la diagonale sont à i pour la tâche i et les autres à 0). Chaque tâche MPI doit envoyer le contenu de cette diagonale à la tâche de rang 0 qui place le contenu de la diagonale de la tâche i dans la rangée i de sa matrice. Le contenu de la matrice est ensuite imprimé par la tâche 0.