UNIVERSITÉ D'ORLÉANS **M1 Info**
Informatique Année **2012 - 2013**

Contrôle continu : Calcul Intensif

Exercice 1. PRAM : encore un calcul de tri

Le tri par comptage consiste à trier les éléments d'un tableau d'entiers en comptant le nombre d'occurrences de chaque valeur contenue dans ce tableau. On suppose T un tableau de n valeurs à trier avec $0 \leq T[i] \leq MAX$ les deux étapes de l'algorithme sont alors les suivantes :

1. Construction du tableau C de taille $MAX + 1$ tel que $C[i]$ est le nombre de fois où l'entier i apparaît dans le tableau,
2. Construction du tableau trié TT en parcourant le tableau C et en recopiant $C[i]$ fois i dans TT.

Par exemple si `T={0,5,7,8,5,1,2,1}` alors `C={1,2,1,0,0,2,0,1,1}` et pour construire TT on copie une fois la valeur 0 puis 2 fois la valeur 1 et ainsi de suite pour trouver `{0,1,1,2,5,5,7,8}`.

Donnez un algorithme PRAM correspondant à cet algorithme en précisant le nombre de processeurs utilisés et la notation permettant d'identifier le numéro de processeur dans votre algorithme. On suppose MAX connu. De plus vous pouvez utiliser la fonction de `reduction` et la fonction `scan` pour le calcul des préfixes sans les écrire. Par contre utilisez des notations claires pour indiquer les processeurs impliqués dans ces fonctions la variable sur laquelle la fonction s'applique ainsi que l'opération utilisée. Par exemple si vous utilisez (i, j) pour identifier un processeur `reduction(+,tab,(i,j) i=0, 0<j<...)` désigne la réduction de la première ligne de `tab`.

Indiquez bien sûr votre complexité finale et le type de machine PRAM sur lequel s'exécute votre algorithme.

Exercice 2. Communications en diagonale

A partir du communicateur `MPI_COMM_WORLD` on souhaite construire des fonctions de communications pour une topologie $N \times N$ de processeurs. Ainsi si on accède au numéro du processeur par

```
MPI_Comm_rank(MPI_COMM_WORLD, &i ) ;
MPI_Comm_size(MPI_COMM_WORLD, &nprocs ) ;
```

alors $nprocs = N \times N$ et dans la topologie $N \times N$ ce processeur est identifié par $(\frac{i}{N}, i\%N)$. On souhaite définir des schémas de communications sur des processeurs appartenant à une même diagonale de la grille "virtuelle" de processeurs. A partir de i le numéro du processeur on est capable de déterminer à quelle diagonale (de haut en bas et de gauche à droite) de la grille virtuelle il appartient soit $diag(i) = i\%N - \frac{i}{N}$.

- Ecrire la fonction qui permet de diffuser (proche de MPI_Bcast) un buffer à l'ensemble des processeurs de sa diagonale. La signature de la fonction sera

  `void MPI_Diffusion(void *sendbuf, int n, MPI_Datatype datatype, int root)`

  avec `sendbuf` le buffer respectivement des valeurs à diffuser par le processeur $root$ ou des valeurs à recevoir pour les autres processeurs de la diagonale, n le nombre d'éléments du tableau et `datatype` le type des éléments de `sendbuf`.

- Ecrire la fonction qui permet de rassembler (proche de MPI_Gather) un ensemble de valeurs des processeurs d'une diagonale sur un processeur donné. La signature de cette fonction sera

  `void MPI_Rassemble(void *sendbuf, int sendcnt, void *recvbuf, int recvcnt, MPI_Datatype datatype, int root)`

  où `sendbuffer` est le buffer de données à envoyer, `sendcnt` le nombre de valeurs envoyées, `recvbuf` le buffer de données pour la réception, `recvcnt` le nombre de valeurs reçues, `datatype` le type des éléments de `sendbuf` et `root` le processeur sur lequel rassembler les valeurs.

- Ecrire la fonction qui permet de faire un all-to-all toujours sur une diagonale de processeurs. La signature de cette fonction similaire à MPI_Alltoall sera

  `MPI_tousatous(void *sendbuf, int sendcount, void *recvbuf, int recvcount, MPI_Datatype recvtype, int ndiag)`

  avec le même sens des paramètres que précédemment plus `ndiag` le numéro de la diagonale de processeurs qui participent à la communication.

Au niveau du programme principal l'appel à une telle fonction pourra se faire de la manière suivante si on note `ndiag` le numéro de la diagonale concernée

```
if (diag(i)==ndiag)
  MPI_Diffusion(tab,n,MPI_INT,root)
if (diag(i)==ndiag)
  MPI_Rassemble(tab1,n1,tab2,n2,MPI_INT,5)
if (diag(i)==ndiag)
  MPI_Toutatous(tab1,n1,tab2,n2,MPI_INT,ndiag)
```