UNIVERSITÉ D'ORLÉANS **M1 Info**
Informatique **2015 - 2016**

Session 1: Programmation Parallèle - 2h

Exercice 1. QuickSort (5pts)

Décrivez sans donner l'implémentation l'algorithme de tri QuickSort pour le modèle PRAM. Vous pourrez illustrer cette description à partir d'un exemple "idéal" avec $n = 16$.

Exercice 2. Suite de Syracuse Version 2 (7pts)

Une suite de Syracuse est une suite telle que $U_0 = x$ avec $x > 0$ et

$$U_n = \begin{cases} \frac{U_{n-1}}{2} & si\ U_{n-1}\ est\ pair \\ 3U_{n-1} + 1 & sinon \end{cases}$$

On souhaite vérifier si un tableau d'entiers de taille N correspond à une suite de Syracuse sachant que le processeur `root` donné en argument du programme génère cette suite de longueur N également donné en paramètre du programme. Mais vous ne devez utiliser que des communications RMA (one-sided communications) pour les échanges entre processeurs.

1. Explicitez les fenêtres dont vous aurez besoin ainsi que le principe de la parallélisation sachant que le processeur `root` est le seul à devoir connaître le résultat final.

2. Complétez le code ci-dessous avec l'implémentation du test de Syracuse en RMA. Les lignes de code pour la fenêtre `win` sont là à titre d'exemple pour vous rappeler la syntaxe de création d'une fenêtre.

```
int main ( int argc , char **argv )
{
  int pid, nprocs;
  MPI_Init (&argc , &argv) ;
  MPI_Comm_rank(MPI_COMM_WORLD, &pid ) ;
  MPI_Comm_size (MPI_COMM_WORLD, &nprocs ) ;

  int N = atoi(argv[1]);
  int root = atoi(argv[2]);

  int* Suite;
  if (pid==root) {
    Suite = new int[N];
    Make_Suite(Suite, N); // Make_Suite est une fonction permettant de générer une suite
  }
  MPI_Win win;
  MPI_Win_create(buf??, size??, sizeof(??),MPI_INFO_NULL,MPI_COMM_WORLD,&win);
  MPI_Win_fence(0,win);

  ......

  MPI_Win_fence(0,win);
  MPI_Win_free(&win);

  if (pid==root)
    delete[] Suite;
  MPI_Finalize() ;
  return 0 ;
}
```

3. On souhaite faire ce test en boucle. Désormais, le processeur `root` génère les suites à tester les unes après les autres et affiche le résultat au fur et à mesure qu'il le recoit. Expliquez le modèle de parallélisme que vous utiliseriez et le principe de la parallélisation correspondante. Aucun code n'est demandé ici juste des descriptions à l'aide d'un schéma éventuellement.

Exercice 3. Une matrice découpée en blocs (8pts)

Cet exercice s'inspire du projet. Il s'agit de travailler sur une matrice A découpée par bloc sur une grille de processeurs. Les hypothèses sont les suivantes

1. vous disposez de $nprocs = p \times p$ processeurs
2. la matrice A est de taille $N \times N$ et se décompose en $n_b \times n_b$ blocs de taille $n \times n$ avec N divisible par n_b donc $n = \frac{N}{n_b}$
3. n_b est divisible par p

On suppose qu'initialement un processeur s'est chargé de distribuer la matrice A sur la grille de processeurs selon une distribution de type round-robin (fonction `distributionN_to_n` du projet).

- Les communications
  1. Ecrivez le code permettant de créer la topologie cartésienne à partir des $nprocs$ processeurs disponibles.
  2. Donnez le calcul qui permet de définir le nombre de blocs $nbblocs$ qu'un processeur doit gérer. Vous noterez `blocs_local` la variable permettant de stocker les A_{ij} reçu par un processeur (`float*` blocs_local = new float[nbblocs*n*n]).
  3. Donnez le calcul qui permet de connaître le numéro du bloc correspondant au bloc A_{ij}. Si un processeur gère $nbblocs$ il s'agit de trouver $0 \leq num \leq nbblocs$ tel que le bloc num en local correspond au bloc A_{ij} de la matrice.
  4. Donnez les règles de calculs et/ou les appels de fonction qui permettent de connaître pour un bloc A_{ij} les coordonnées (k, l) du processeur qui a reçu ce bloc A_{ij} puis son rang $rank$.
  5. Définissez la signature de la fonction qui permet de diffuser le bloc A_{i0} à tous les processeurs ayant déjà un bloc A_{ix} de la matrice A et donnez le code de son implémentation.
  6. Généralisez la fonction précédente (en modifiant sa signature si nécessaire) afin que le processeur gérant le bloc A_{ij} le diffuse à tous les processeurs ayant un bloc A_{ix}.
- Dans le cadre de la factorisation LU étudiée lors du projet
  1. Estimez la complexité obtenue par cette parallélisation.
  2. Y aurait-il un intérêt à utiliser des communications RMA ? Justifiez.