EISTI	EX - Examen n°1
Rachid Chelouah - Juan Angel Lorenzo	Architecture et programmation parallèle et répartie
ING2-GSI-MI	Année 2015–2016

Modalités

- Durée : 2 heures.
- Vous devez rédiger votre copie à l'aide d'un stylo à encre exclusivement.
- Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
- Aucun document n'est autorisé.
- Aucune question ne peut être posée aux enseignants, posez des hypothèses en cas de doute.
- Aucune machine électronique ne doit se trouver sur vous ou à proximité, même éteinte.
- Aucune sortie n'est autorisée avant une durée incompressible d'une heure.
- Aucun déplacement n'est autorisé.
- Aucun échange, de quelque nature que ce soit, n'est possible.

Questions courtes

(1) On a récupéré le code ci-dessous qui initialise les éléments d'un tableau par le carré de i et l'affiche, puis par le cube de i. Ce code donne-t-il le résultat voulu ? Sinon que faut-il ajouter et à quel endroit ? □

```
1  #include <omp.h>
2  #include <iostream>
3  using namespace std;
4
5  int main(int argc, char* argv[])  {
6     int a[5], i;
7     #pragma omp parallel
8     {
9         #pragma omp for
10         for (i = 0; i < 5; i++)
11             a[i] = i * i;
12        #pragma omp master
13         for (i = 0; i < 5; i++)
14             cout << "a[" << i << "] = " << a[i] << endl;
15        #pragma omp for
16         for (i = 0; i < 5; i++)
17            a[i] = i * i * i ;
18     }
19     return 0;
20 }
```

(2) Étant donné un tableau T d'entiers de taille $taille$, on veut le trier en utilisant la version récursive de l'algorithme de tri fusion ci-dessous. Quelle est la façon la plus efficace de paralléliser cet algorithme en OpenMP ? Expliquez brièvement. □

```
1  void tri (int T[], int size){
2     tri(T, taille/2);
3     tri(T+taille/2, taille - taille/2);
4     fusion(T, taille);
5  }
```

Exercice 1 : jeu de dés

On vous propose d'implémenter un jeu de dés à plusieurs joueurs. Le maitre donne le top de départ de la partie. Chaque joueur est indépendant, il lance les dés et comptabilise son nombre de points. Le premier qui atteint un nombre défini de points, on arrête la partie. Le processus maître affiche le numéro du joueur (processus) gagnant et la somme cumulée de points.

1. Proposer et implémenter un programme parallèle MPI point à point
2. Proposer et implémenter un programme parallèle en utilisant l'API OpenMP
3. Dans ce dernier cas, donner les différentes manières de synchroniser le calcul de la somme cumulée des points

Exercice 2 : nombres premiers

On suppose un tableau T d'entiers de taille L rempli aléatoirement de nombre inférieur à N. On veut calculer le nombre de nombres premiers contenus dans ce tableau. Nous avons à notre disposition une fonction `isPrime(m)` qui retourne 1 si m est premier sinon elle retourne un 0. On suppose que nous avons P processus, et on veut paralléliser ce traitement. Écrivez le code MPI en utilisant la communication point à point.

Exercice 3 : communications collectives

Soit un programme `mpi_allreduce.cc` qui, en utilisant la primitive `MPI::Comm::Allreduce`, fait une réduction répartie du calcul du produit des rangs des processus sauf pour le processus P0 qui est fixé à 10 :

```
void MPI::Comm::Allreduce(const void* sendbuf, void* recvbuf, int count,
                          const MPI::Datatype& datatype, const MPI::Op& op)
```

1. Expliquer la primitive `MPI::Comm::Allreduce`
2. En quelles fonctions point-à-point on peut l'assimiler (équivalence) ?
3. Donner le code de ce programme
4. Quelle est la commande pour exécuter ce programme avec 5 processus ?
5. Donner la sortie de cette exécution