Corrigé type du contrôle de CP

Questions du Cours : (04 points)

Q-1) Expliquer le principe de la programmation MPMD ? **(01 points)**

R-1) Dans ce modèle de programmation les processus sont obtenus des codes ou programme différents et s'exécutent en traitant des données différentes.

Q-2) Quelle la différence entre une application parallèle et une application concurrente ? **(01 points)**

R-2) Le premier cas, présente l'exécution simultané des calculs, autant que le second cas, présente la composition des exécutions indépendantes des processus.

Q-3) Pourquoi on utilise deux classements (SPECint et SPECfp) dans le SPECtest ? **(01 points)**

R-3) On utilise les deux classements (SPECint et SPECfp) dans le SPECtest parceque ce sont les deux modes de calcul de n'importe quel processeur.

Q-4) Citer et expliquer les différentes phases utilisées par un programme parallèle ? **(01 points)**

R-4) Citer et expliquer les différentes phases utilisées par un programme parallèle ? **(01 points)**

Programme principale
…
Préparation des données
…
Création et initialisation des processus

Processus N° 1

1) Réception des données à traiter;
2) Traitement des données;
3) Envoi des résultats;

Processus N° 2

1) Réception des données à traiter;
2) Traitement des données;
3) Envoi des résultats;

…

Processus N° k

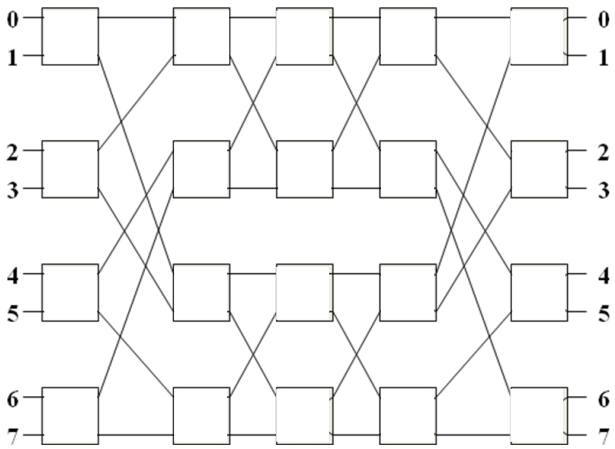
1) Réception des données à traiter;
2) Traitement des données;
3) Envoi des résultats;

Récupération des résultats
…
Affichage des résultats

Exercice N° 01: (06 points)

Créer un réseau Benes de 8x8, ensuite, appliquer la matrice ci-dessous sur ces réseaux :

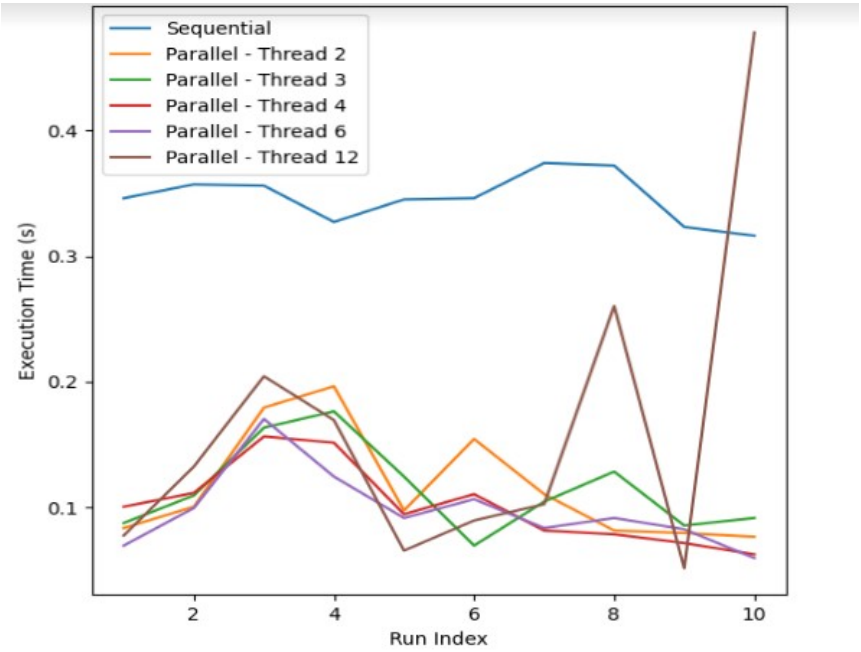
$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 5 & 7 & 3 & 2 & 6 & 0 & 4 \end{pmatrix}$$



Appliquer l'algorithme de BENES pour fixer les états des commutateurs.

Exercice Nº 02: (06 points)

Soit la figure ci-dessous qui représente les valeurs des temps d'exécution en millisecondes de six versions des programmes qui permettent de compresser un ensemble de fichiers de taille inférieur à 10 kilobytes, avec un nombre de threads différent à chaque fois.



Q-1) Pourquoi les valeurs d'exécution de la même version du programme se diffère d'une exécution à l'autre ? **(02 points)**

R-1) Le changement dans le temps d'exécution est dû à la variation de temps de latence (temps d'attente) pour exécuter un processus à cause de l'exécution concurrente.

Q-2) Donnez vos remarques sur les différentes courbes représentés par la figure ci-dessus ? **(04 points)**

R-2) Nous pouvons remarquer le suivant :

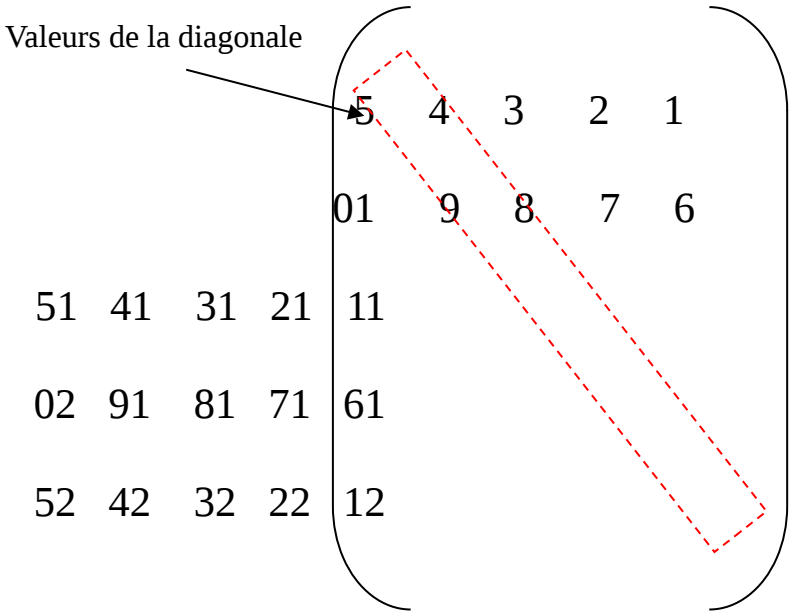
- L'utilisation d'un ensemble de threads permet de gagner un temps important de traitement.
- L'ajout de threads pour une exécution parallèle est limité par la capacité de la mémoire et d'exécution de la machine.

Exercice Nº 03 : (04 points)

Donnez le programme Python utilisant la bibliothèque MPI, qui suit le scénario suivant :

- Le processus 0 prépare une matrice M (initialisation de la matrice avec des valeurs aléatoires), ensuite, envoi la matrice aux différents processus, et finalement attend le résultat de chaque processus pour afficher les valeurs min, max et somme.

- Le processus 1 calcule la valeur minimale (min) des valeurs au-dessous de la diagonale de la matrice M.
- Le processus 2 calcule la valeur maximale (max) des valeurs au-dessus de la diagonale de la matrice M.
- Le processus 3 calcule la somme des valeurs de la diagonale de la matrice M.




```
import numpy as np
import random
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

if rank == 0:
    print('process 0 start processing...')
    upper = []
    lower = []
    N = 5
    x = np.zeros((N, N))
    for i in range(0, N):
        for j in range(0, N):
            x[i, j] = random.randint(0, 100)
    print('The value is :\n', x)

    for d in range(0, N):
```

```
        for k in range(d+1, N):
            upper.append(x[d, k])
            lower.append(x[k, d])

    diag = x.diagonal()

    # Send data to processes

    comm.send(lower, dest=1, tag=1)
    comm.send(upper, dest=2, tag=1)
    comm.send(diag, dest=3, tag=1)

    minval = comm.recv(source=1, tag=1)
    maxval = comm.recv(source=2, tag=1)
    som= comm.recv(source=3, tag=1)

    print('somme = ', som)
    print('min= ', minval)
    print('max= ', maxval)
    print('process 0  finished...')
if rank == 1:
    data = comm.recv(source=0, tag=1)
    minval = min(data)
    comm.send(minval, dest=0, tag=1)
    print('process 1 finished...')

if rank == 2:
    data = comm.recv(source=0, tag=1)
    maxval = max(data)
    comm.send(maxval, dest=0, tag=1)
    print('process 2 finished...')

if rank == 3:
    data = comm.recv(source=0, tag=1)
    som = sum(data)
    comm.send(som, dest=0, tag=1)
    print('process 2 receving data finished...')
```

Bon Courage **Bonne Chance**