Exercice à rendre

2.1 Question du cours

1&2 il y a six situations d'exécutions totalement qui sont

012	**021**	**102**	**120**	**201**	**210**
x	x	>	>	x	>

x *représent l'exécution normale, et* ***>*** *représent l'exécution d'interblocage.*

Parce que chaque situation se produit avec la même probabilité, la probabilité de d'avoir un interblocage est 0.5.

2.2 Question du cours n°2

1.loi d'Amdhal = n/(1+(n-1)f), et f égal à 0.1(1-90%).
Donc loi d'Amdhal S= 10/(1+9/n)Lorsque n tend vers l'infini, S est égal à 10.
2.On suppose s≥9, c'est-à-dire 10/(1+9/n),alors on peut obtenir n≥81.
3.Parce que s est égal à 4. on peut calculer que n utilisé par Alice est 6(s=10/(1+9/n)=4).De plus, on sait que Ss (n) = s + n.p(loi de Gustafson), où s représente la proportion en temps du code exécuté en séquentiel, donc s=0.1.
Par conséquence Ss(n)=6+(1-6)×0.1=5.5

2.3 Ensemble de mandelbrot

Quand j'exécute le programme avec 8 tâches(mpirun -np 8 ./Mandelbrot_mpi.exe)**, il affiche que:**
[0] Temps calcul ensemble mandelbrot : 0.23963
[7] Temps calcul ensemble mandelbrot : 0.241013
[3] Temps calcul ensemble mandelbrot : 1.90915
[4] Temps calcul ensemble mandelbrot : 1.93
[2] Temps calcul ensemble mandelbrot : 1.94582
[5] Temps calcul ensemble mandelbrot : 1.96247
[1] Temps calcul ensemble mandelbrot : 5.15961
[6] Temps calcul ensemble mandelbrot : 5.17626
temps total :5.20392s

cependant, lorsque j'exécute le programme avec une tâche, le temps pour calculer mandabrot est:
Temps calcul ensemble mandelbrot : 16.1205
temps total :16.1683s

Par conséquence, speedup S(8) = ts/tp(8)= 16.17/5.20= 3.11

Dans cette code, Je divise la *Height* de l'image avec équilibre en fonction du nombre de tâches. Ensuite, je utilise la fonction MPI_Gather pour composer les données de tâches. Bien que la nombre de pixels calculé par chaque tâche soit le même, le montant du calcul est différent, ce qui entraîne un déséquilibre du calcul(comme tâche 1 et 6).

Lorsque que je utilise la stratégie de naître-esclave, le temps total est temps total :2.45398s. Pour bien comparer les résultats de les deux methodes, j'exécute la programme avec les même tâche(mpirun -np 8 ./MandelbrotEsclave.exe)

Par conséquence, speedup S(8)=ts/tp(8)=16.17/2.45=6.6

temps total :2.45398s

conclusion: ici, je divise la mission de calcule d'image en 200 petites sub-tâches. Cette stratégie évite la gaspillage de ressources causé par des threads à exécution rapide en attente d'autres threads.

2.4 Produit matrice–vecteur

Je utilise les deux fontions MPI_Scatter et MPI_Gather pour terminer les tâches. Veuillez vérifier le code correspondant.