Examen IN203

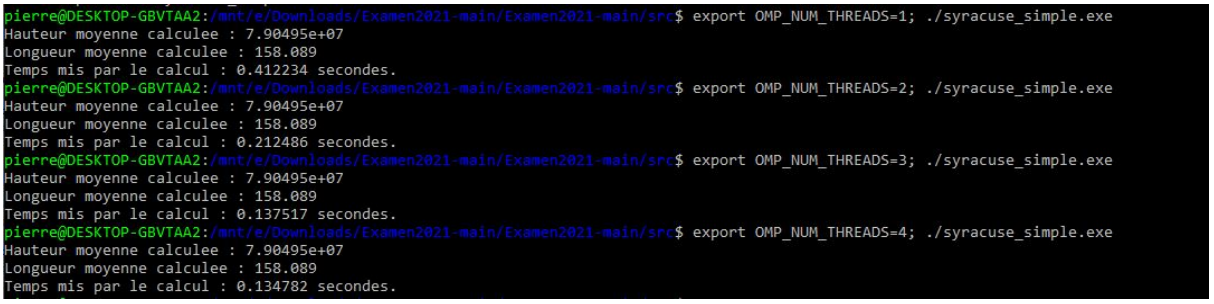
2. Introduction

Mon CPU à 4 coeurs et avec 1 thread par cœur soit 4 threads. J'ai une mémoire cache L2 de 1,0 Mo et une mémoire cache L3 de 6,0 Mo.

3. Suite de Syracuse

Syracuse simple

Un calcul initial de la suite de Syracuse sans parallélisation me donne une hauteur moyenne de 7.90495e7 avec une longueur moyenne de 158.089 et un temps mis par le calcul de 0.369097s. En parallélisant le programme j'obtiens ces résultats :



Soit un speed-up de 90% pour 1 thread, 176% pour 2, 270% pour 3 et 276% pour 4. On a un speed up qui augmente d'abord fortement car on diminue le nombre d'itérations que le processeur doit faire séquentiellement en divisant le travail entre les threads. À partir de 3 threads on voit que le temps de calcul augmente beaucoup moins car il y a une partie du code du début qui est séquentielle et qui n'est donc pas affectée par notre parallélisation du code.

Syracuse avec orbite

L'exécution du programme sans parallélisation nous donne un temps total d'exécution de 1.52s et on obtient les résultats suivant en parallélisant la boucle:

Ce qui donne des speed-up 102% pour 1 thread, 109% pour 2, 111% pour 3 et 106% pour 4. Ce speed-up est beaucoup moins bon que pour la suite de Syracuse simple et peut s'expliquer par le fait que la suite simple est cpu bound quand celle avec orbite est memory bound. Ainsi on ne ressent pas d'accélération réelle dans le 2ème cas car la vitesse du programme est limitée par l'accès à la mémoire (quand on conserve les valeurs des suites dans le tableau) et on ne gagne pas vraiment de performances en divisant le travail dans le processeur.

Simple automate cellulaire

Parallélisation OpenMP

Pour l'exécution de notre programme en version séquentielle on a un temps de calcul de 0.71s et un temps pour constituer les images de 0.39s.
En parallélisant notre programme on obtient les résultats suivant:

```
pierre@DESKTOP-GBVTAA2:/mnt/e/Downloads/Examen2021-main/Examen2021-main/src$ export OMP_NUM_THREADS=1; ./simple_automate_cellulaire.exe
Resume des parametres :
        Nombre de cellules  : 1000
        Nombre d'iterations : 1000
        Degre de voisinage  : 1
        Nombre de cas       : 256

Temps mis par le calcul : 0.811748 secondes.
Temps mis pour constituer les images : 0.404803 secondes.
pierre@DESKTOP-GBVTAA2:/mnt/e/Downloads/Examen2021-main/Examen2021-main/src$ export OMP_NUM_THREADS=2; ./simple_automate_cellulaire.exe
Resume des parametres :
        Nombre de cellules  : 1000
        Nombre d'iterations : 1000
        Degre de voisinage  : 1
        Nombre de cas       : 256

Temps mis par le calcul : 0.830022 secondes.
Temps mis pour constituer les images : 0.429242 secondes.
pierre@DESKTOP-GBVTAA2:/mnt/e/Downloads/Examen2021-main/Examen2021-main/src$ export OMP_NUM_THREADS=3; ./simple_automate_cellulaire.exe
Resume des parametres :
        Nombre de cellules  : 1000
        Nombre d'iterations : 1000
        Degre de voisinage  : 1
        Nombre de cas       : 256

Temps mis par le calcul : 0.816419 secondes.
Temps mis pour constituer les images : 0.432832 secondes.
pierre@DESKTOP-GBVTAA2:/mnt/e/Downloads/Examen2021-main/Examen2021-main/src$ export OMP_NUM_THREADS=4; ./simple_automate_cellulaire.exe
Resume des parametres :
        Nombre de cellules  : 1000
        Nombre d'iterations : 1000
        Degre de voisinage  : 1
        Nombre de cas       : 256

Temps mis par le calcul : 0.854526 secondes.
Temps mis pour constituer les images : 0.549745 secondes.
```

On a donc de moins bonnes performances car on a:
-pour 1 thread un speed-up de 88% en calcul pour un temps équivalent en image.
-pour 2 threads un speed-up de 85% en calcul et 93% en image.
-pour 3 threads un speed-up de 88% en calcul et 91% en image.
-pour 4 threads un speed-up de 84% en calcul en 71% en image.

Ces performances font sens car il y a énormément d'accès mémoire dans la liste cell au cours du programme et assez peu de calculs. Ainsi on observe pas de speed-up en parallélisant les calculs faits par le processeur.