Rapport – Travaux Pratique 7

Sommaire

[Exercice 1 1](#_Toc98005264)

[Exercice 2 2](#_Toc98005265)

[Exercice 3 3](#_Toc98005266)

[Exercice 4 4](#_Toc98005267)

[Exercice 5 5](#_Toc98005268)

Ce TP a été réalisé avec un GPU Tesla K80

# Exercice 1

|  |  |  |
| --- | --- | --- |
| Taille des tableaux | Moyenne du temps d’exécution du patron REDUCE sur GPU (µs) | Moyenne du temps d’exécution du patron REDUCE sur CPU (µs) |
| 600 x 450 = 270 000 | 245 | 1 174 |
| 2 024 x 2 657 = 5 377 768 | 4 054 | 23 913 |
| 4 460 x 2 974 = 13 264 040 | 9 776 | 82 226 |

Plus le tableau contient de valeurs, plus le temps d’exécution sur le GPU est rapide par rapport au temps d’exécution sur le CPU. Pour un tableau de plus de 13 millions de valeurs, le temps d’exécution sur GPU est 8.5 fois plus rapide que le temps d’exécution sur CPU.

# Exercice 2

|  |  |
| --- | --- |
| Taille des tableaux / Nombre de warps | Moyenne du temps d’exécution du patron REDUCE sur GPU (µs) |
| 270 000 / 1 | 329 |
| 270 000 / 2 | 184 |
| 270 000 / 4 | 158 |
| 270 000 / 8 | 109 |
| 270 000 / 16 | 139 |
| 270 000 / 32 | 261 |
|  | |
| 5 377 768 / 1 | 5 023 |
| 5 377 768 / 2 | 2 702 |
| 5 377 768 / 4 | 2 302 |
| 5 377 768 / 8 | 1 376 |
| 5 377 768 / 16 | 2 045 |
| 5 377 768 / 32 | 4 365 |
|  | |
| 13 264 040 / 1 | 12 043 |
| 13 264 040 / 2 | 6 483 |
| 13 264 040 / 4 | 5 515 |
| 13 264 040 / 8 | 3 276 |
| 13 264 040 / 16 | 4 904 |
| 13 264 040 / 32 | 10 530 |

On remarque que le meilleur découpage en termes de temps d’exécution se fait avec 8 warps, ce qui représente 256 threads. De plus on voit bien qu’en réduisant la pression sur les registres notre patron REDUCE est beaucoup plus rapide que celui réalisé dans l’exercice précédent. Ainsi avec 8 warps on est 20 fois rapide sur un tableau de 13 millions de valeurs.

# Exercice 3

|  |  |
| --- | --- |
| Taille des tableaux / Nombre de warps | Moyenne du temps d’exécution du patron REDUCE sur GPU (µs) |
| 270 000 / 1 | 92 |
| 270 000 / 2 | 83 |
| 270 000 / 4 | 77 |
| 270 000 / 8 | 92 |
| 270 000 / 16 | 144 |
| 270 000 / 32 | 273 |
|  | |
| 5 377 768 / 1 | 1 064 |
| 5 377 768 / 2 | 987 |
| 5 377 768 / 4 | 900 |
| 5 377 768 / 8 | 1 066 |
| 5 377 768 / 16 | 2 124 |
| 5 377 768 / 32 | 4 574 |
|  | |
| 13 264 040 / 1 | 2 538 |
| 13 264 040 / 2 | 2 337 |
| 13 264 040 / 4 | 2 156 |
| 13 264 040 / 8 | 2 530 |
| 13 264 040 / 16 | 5 101 |
| 13 264 040 / 32 | 11 038 |

Dans cette version qui utilise la propriété de commutativité, il faut cette fois-ci utiliser 4 warps, donc 128 threads, pour avoir le meilleur temps d’exécution. On remarque qu’on est plus rapide que dans l’exercice précédents.

# Exercice 4

|  |  |
| --- | --- |
| Taille des tableaux / Nombre de warps | Moyenne du temps d’exécution du patron REDUCE sur GPU (µs) |
| 270 000 / 1 | 91 |
| 270 000 / 2 | 82 |
| 270 000 / 4 | 76 |
| 270 000 / 8 | 91 |
| 270 000 / 16 | 142 |
| 270 000 / 32 | 271 |
|  | |
| 5 377 768 / 1 | 1 059 |
| 5 377 768 / 2 | 984 |
| 5 377 768 / 4 | 899 |
| 5 377 768 / 8 | 1 053 |
| 5 377 768 / 16 | 2 095 |
| 5 377 768 / 32 | 4 566 |
|  | |
| 13 264 040 / 1 | 2 531 |
| 13 264 040 / 2 | 2 334 |
| 13 264 040 / 4 | 2 157 |
| 13 264 040 / 8 | 2 495 |
| 13 264 040 / 16 | 5 030 |
| 13 264 040 / 32 | 11 018 |

Les temps d’exécution sont semblables à l’exercice précédent avec quelques µs en moins.

# Exercice 5

|  |  |
| --- | --- |
| Taille des tableaux / Nombre de warps | Moyenne du temps d’exécution du patron REDUCE sur GPU (µs) |
| 270 000 / 1 | 88 |
| 270 000 / 2 | 78 |
| 270 000 / 4 | 71 |
| 270 000 / 8 | 81 |
| 270 000 / 16 | 113 |
| 270 000 / 32 | 203 |
|  | |
| 5 377 768 / 1 | 1 024 |
| 5 377 768 / 2 | 930 |
| 5 377 768 / 4 | 898 |
| 5 377 768 / 8 | 959 |
| 5 377 768 / 16 | 1 557 |
| 5 377 768 / 32 | 3 272 |
|  | |
| 13 264 040 / 1 | 2 444 |
| 13 264 040 / 2 | 2 214 |
| 13 264 040 / 4 | 2 160 |
| 13 264 040 / 8 | 2 273 |
| 13 264 040 / 16 | 3 731 |
| 13 264 040 / 32 | 7 883 |

Dans cette dernière version du patron REDUCE on est légèrement plus rapide que l’exercice précédent quand le nombre de warps est inférieur à 4, au-dessus il y a plus de threads et on observe donc mieux l’optimisation.