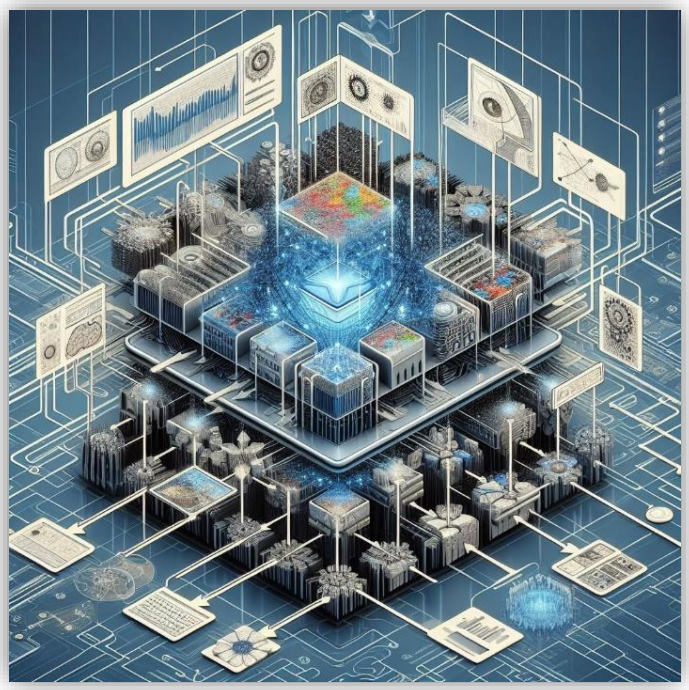

Sommaire

Spécifications de ma machine	
Processor	AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx 2.30 GHz
Installed RAM	16.0 GB (13.9 GB usable)
System type	64-bit operating system, x64-based processor
Windows	Windows 11 Home
GPU Device	NVIDIA GeForce GTX 1650

GPU-Total amount of global memory:	4096 MBytes (4 294 639 616 bytes)
GPU-Max dimension size of a grid size (x,y,z):	(2 147 483 647, 65 535, 65 535)
GPU-Max dimension size of a thread block (x,y,z):	(1024, 1024, 64)
GPU-Total number of registers available per block:	65 536
GPU-Warp size:	32

Exercice 1

Voici les résultats obtenus en appliquant l'effet effet par bloc :

Figure : Image wraps-avocat-houmous-et-crevettes.ppm avant l'application de l'effet par bloc

Figure : Image wraps-avocat-houmous-et-crevettes_student.ppm après l'application de l'effet par bloc

Fondamentalement, dans le résultat des prochaines questions, il n'y aura pas de différence. D'où le fait de mettre les résultats que dans cette question. La différence réside dans la façon d'implémenter le patron et ses performances, C'est ce que nous allons explorer dans les prochaines questions.

Taille des Images	Moyenne du temps d'exécution du patron REDUCE sur GPU (µs)	Moyenne du temps d'exécution du patron REDUCE sur CPU (µs)
CaravaggioUrsula.ppm (289 800 pixels)	178	2 197
Paris.ppm (562 000 pixels)	318	4 624
Nuit.ppm (1 764 000 pixels)	925	13 899
MonSalon.ppm (10 036 224 pixels)	4 988	81 459

On observe que le temps d'exécution du patron REDUCE sur GPU est nettement inférieur à celui sur CPU pour toutes les tailles d'images testées. Cette amélioration du temps au niveau du GPU est d'autant plus marquée que la taille de l'image est grande.

Pour l'image CaravaggioUrsula.ppm (289 800 pixels), le temps d'exécution sur GPU est de 178 µs, contre 2 197 µs sur CPU. On constate donc une accélération d'un facteur d'environ 12.

Pour l'image Paris.ppm (562 000 pixels), le temps d'exécution sur GPU est de 318 µs, contre 4 624 µs sur CPU. L'accélération est ici d'un facteur d'environ 14.

Pour l'image Nuit.ppm (1 764 000 pixels), le temps d'exécution sur GPU est de 925 µs, contre 13 899 µs sur CPU. L'accélération est ici d'un facteur d'environ 15.

Enfin, pour l'image MonSalon.ppm (10 036 224 pixels), le temps d'exécution sur GPU est de 4 988 µs, contre 81 459 µs sur CPU. L'accélération est ici d'un facteur d'environ 16.

Exercice 2

Taille des Images / Nombre de warps	Moyenne du temps d'exécution du patron REDUCE sur GPU (µs)
CaravaggioUrsula.ppm (289 800 pixels) / 1	575
(289 800 pixels) / 2	547
(289 800 pixels) / 4	467
(289 800 pixels) / 8	493
(289 800 pixels) / 16	552
(289 800 pixels) / 32	666
Paris.ppm (562 000 pixels) / 1	621
(562 000 pixels) / 2	515
(562 000 pixels) / 4	441
(562 000 pixels) / 8	517
(562 000 pixels) / 16	557
(562 000 pixels) / 32	661
Nuit.ppm (1 764 000 pixels) / 1	634
(1 764 000 pixels) / 2	545
(1 764 000 pixels) / 4	429
(1 764 000 pixels) / 8	450
(1 764 000 pixels) / 16	539
(1 764 000 pixels) / 32	900
MonSalon.ppm (10 036 224 pixels) / 1	3 110
(10 036 224 pixels) / 2	2 309
(10 036 224 pixels) / 4	2 150
(10 036 224 pixels) / 8	2 311
(10 036 224 pixels) / 16	2 316
(10 036 224 pixels) / 32	4 976

On observe que le temps d'exécution du patron varie en fonction du nombre de warps utilisés par bloc. Pour toutes les images testées, on constate que le temps d'exécution augmente lorsque le nombre de warps par bloc est soit très faible (1 warp), soit très élevé (32 warps).

Pour chaque image, on trouve un "point optimal" où le temps d'exécution est le plus bas. Par exemple, pour l'image CaravaggioUrsula.ppm (289 800 pixels), le temps d'exécution est le plus bas avec 4 warps par bloc (467 µs). De même, pour l'image Paris.ppm (562 000 pixels), l'image Nuit.ppm (1 764 000 pixels) et l'image MonSalon.ppm (10 036 224 pixels), le temps d'exécution est le plus bas avec respectivement 4, 4 et 4 warps par bloc.

Les résultats montrent que l'utilisation d'un nombre modéré de warps par bloc peut aider à optimiser l'efficacité du patron REDUCE sur GPU. Cela est probablement dû au fait qu'un nombre trop élevé de warps par bloc peut entraîner une saturation des ressources du GPU, tandis qu'un nombre trop faible de warps par bloc peut ne pas exploiter pleinement le parallélisme offert par le GPU.

Exercice 3

Taille des Images / Nombre de warps	Moyenne du temps d'exécution du patron REDUCE sur GPU (µs)
CaravaggioUrsula.ppm (289 800 pixels) / 1	577
(289 800 pixels) / 2	520
(289 800 pixels) / 4	503
(289 800 pixels) / 8	513
(289 800 pixels) / 16	563
(289 800 pixels) / 32	639
Paris.ppm (562 000 pixels) / 1	574
(562 000 pixels) / 2	502
(562 000 pixels) / 4	441
(562 000 pixels) / 8	463
(562 000 pixels) / 16	513
(562 000 pixels) / 32	579
Nuit.ppm (1 764 000 pixels) / 1	513
(1 764 000 pixels) / 2	487
(1 764 000 pixels) / 4	438
(1 764 000 pixels) / 8	463
(1 764 000 pixels) / 16	487
(1 764 000 pixels) / 32	942
MonSalon.ppm (10 036 224 pixels) / 1	2 253
(10 036 224 pixels) / 2	2 247
(10 036 224 pixels) / 4	2 200
(10 036 224 pixels) / 8	2 299
(10 036 224 pixels) / 16	2 338
(10 036 224 pixels) / 32	5 179

On observe que le temps d'exécution du patron varie en fonction du nombre de warps utilisés par bloc. Pour toutes les images testées, on constate que le temps d'exécution augmente lorsque le nombre de warps par bloc est soit très faible (1 warp), soit très élevé (32 warps).

Pour chaque image, on trouve un "point optimal" où le temps d'exécution est le plus bas. Par exemple, pour l'image CaravaggioUrsula.ppm (289 800 pixels), le temps d'exécution est le plus bas avec 4 warps par bloc (503 µs). De même, pour l'image Paris.ppm (562 000 pixels), l'image Nuit.ppm (1 764 000 pixels) et l'image MonSalon.ppm (10 036 224 pixels), le temps d'exécution est le plus bas avec respectivement 4, 4 et 4 warps par bloc.

Comparé aux résultats de l'exercice précédent, on constate que le temps d'exécution a légèrement augmenté pour certaines images et certains nombres de warps. Cela pourrait être dû à la modification du chargement des données en mémoire cache. Cependant, la différence de temps d'exécution reste relativement faible, ce qui peut suggérer que la modification n'a pas eu d'impact majeur sur les performances du patron REDUCE.

Exercice 4

Taille des Images / Nombre de warps	Moyenne du temps d'exécution du patron REDUCE sur GPU (µs)
CaravaggioUrsula.ppm (289 800 pixels) / 1	97
(289 800 pixels) / 2	87
(289 800 pixels) / 4	85
(289 800 pixels) / 8	92
(289 800 pixels) / 16	123
(289 800 pixels) / 32	183
Paris.ppm (562 000 pixels) / 1	146
(562 000 pixels) / 2	143
(562 000 pixels) / 4	140
(562 000 pixels) / 8	159
(562 000 pixels) / 16	192
(562 000 pixels) / 32	332
Nuit.ppm (1 764 000 pixels) / 1	406
(1 764 000 pixels) / 2	404
(1 764 000 pixels) / 4	400
(1 764 000 pixels) / 8	410
(1 764 000 pixels) / 16	413
(1 764 000 pixels) / 32	920
MonSalon.ppm (10 036 224 pixels) / 1	2 237
(10 036 224 pixels) / 2	2 236
(10 036 224 pixels) / 4	2 232
(10 036 224 pixels) / 8	2 283
(10 036 224 pixels) / 16	2 265
(10 036 224 pixels) / 32	4 844

On observe que le temps d'exécution du patron a considérablement diminué par rapport à l'exercice précédent. Cela montre que la modification du schéma de calcul de la fonction reduceJumpingStep a permis d'accélérer la réduction.

Pour chaque image, on trouve un "point optimal" où le temps d'exécution est le plus bas. Par exemple, pour l'image CaravaggioUrsula.ppm (289 800 pixels), le temps d'exécution est le plus bas avec 4 warps par bloc (85 µs). De même, pour l'image Paris.ppm (562 000 pixels), l'image Nuit.ppm (1 764 000 pixels) et l'image MonSalon.ppm (10 036 224 pixels), le temps d'exécution est le plus bas avec respectivement 4, 4 et 4 warps par bloc.

Comparé aux résultats de l'exercice précédent, on constate que le temps d'exécution a diminué pour toutes les images et tous les nombres de warps. Cela pourrait être dû à la réduction du nombre de warps au travail, ce qui a permis de réduire les latences dues aux synchronisations.

Exercice 5

Taille des Images / Nombre de warps	Moyenne du temps d'exécution du patron REDUCE sur GPU (µs)
CaravaggioUrsula.ppm (289 800 pixels) / 1	94
(289 800 pixels) / 2	89
(289 800 pixels) / 4	99
(289 800 pixels) / 8	129
(289 800 pixels) / 16	141
(289 800 pixels) / 32	146
Paris.ppm (562 000 pixels) / 1	148
(562 000 pixels) / 2	147
(562 000 pixels) / 4	143
(562 000 pixels) / 8	154
(562 000 pixels) / 16	156
(562 000 pixels) / 32	243
Nuit.ppm (1 764 000 pixels) / 1	408
(1 764 000 pixels) / 2	407
(1 764 000 pixels) / 4	406
(1 764 000 pixels) / 8	405
(1 764 000 pixels) / 16	410
(1 764 000 pixels) / 32	684
MonSalon.ppm (10 036 224 pixels) / 1	2 250
(10 036 224 pixels) / 2	2 240
(10 036 224 pixels) / 4	2 237
(10 036 224 pixels) / 8	2 234
(10 036 224 pixels) / 16	2 239
(10 036 224 pixels) / 32	3 594

On observe que le temps d'exécution du patron a légèrement diminué par rapport à l'exercice précédent pour certaines images et certains nombres de warps. Cela suggère que la suppression des synchronisations pour les dernières réductions a permis d'accélérer la réduction.

Pour chaque image, on trouve un "point optimal" où le temps d'exécution est le plus bas. Par exemple, pour l'image CaravaggioUrsula.ppm (289 800 pixels), le temps d'exécution est le plus bas avec 2 warps par bloc (89 µs). De même, pour l'image Paris.ppm (562 000 pixels), l'image Nuit.ppm (1 764 000 pixels) et l'image MonSalon.ppm (10 036 224 pixels), le temps d'exécution est le plus bas avec respectivement 4, 4 et 4 warps par bloc.

Comparé aux résultats de l'exercice précédent, on constate que le temps d'exécution a diminué pour certaines images et certains nombres de warps. Cela pourrait être dû à la suppression des synchronisations pour les dernières réductions, ce qui a permis de réduire les latences dues aux synchronisations.