**수치 컴퓨팅 및 GPU 프로그래밍**

**hw1 report**

20191571 김세영

**1. 올바르게 구현한 기능**

SoA\_GS\_CPU, AoS\_GS\_CPU, SoA\_SO\_CPU, AoS\_SO\_CPU ,

SoA\_GS\_GPU, AoS\_GS\_GPU, SoA\_SO\_GPU, AoS\_SO\_GPU

**2. 환경 명세**

Operation System: Microsoft Windows 11 Education 64-bit

Compiler: Microsoft Visual Studio Community 2019

CPU: Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz 2.90 GHz

GPU: NVIDIA GeForce GTX 1660 SUPER

OpenCL Version: 3.0

**3. 실험 결과**

수행 시간 측정: N\_EXECUTIONS = 10

전체 수행시간/수행 횟수로 평균 수행시간을 구했다. GPU는 kernel execution에서 QUEUED 부터 END까지를 수행시간으로 측정하였다.

밑의 table에서 GPU의 첫 번째 row는 평균 수행 시간 (단위: ms), 두 번째 row는 CPU와 비교 했을 때의 speedup이다.

1. INPUT\_IMAGE = 0 (width = 7360, height = 4832)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Image0 | LWS | (8,16) | (16,16) | (32,16) | (64,16) | (32,32) |
| CPU | SoA\_GS\_CPU | 268.777 | | | | |
| AoS\_GS\_CPU | 275.94 | | | | |
| GPU | SoA\_GS\_GPU | 3.993 | 2.753 | 2.671 | 2.905 | 2.989 |
| 67.31205 | 97.63058 | 100.6279 | 92.5222 | 89.92205 |
| AoS\_GS\_GPU | 3.363 | 2.565 | 2.198 | 2.479 | 2.536 |
| 82.05174 | 107.5789 | 125.5414 | 111.311 | 108.8091 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Image0 | LWS | (8,16) | (16,16) | (32,16) | (64,16) | (32,32) |
| CPU | SoA\_SO\_CPU | 3372.066 | | | | |
| AoS\_SO\_CPU | 3375.525 | | | | |
| GPU | SoA\_SO\_GPU | 8.452 | 8.736 | 10.163 | 15.368 | 15.451 |
| 398.9666 | 385.9966 | 331.7983 | 219.4213 | 218.2426 |
| AoS\_SO\_GPU | 10.367 | 8.545 | 10.092 | 14.363 | 14.16 |
| 325.6029 | 395.0293 | 334.4753 | 235.0153 | 238.3845 |

1. INPUT\_IMAGE = 1(width = 9984, height = 6400)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Image1 | LWS | (8,16) | (16,16) | (32,16) | (64,16) | (32,32) |
| CPU | SoA\_GS\_CPU | 505.518 | | | | |
| AoS\_GS\_CPU | 490.401 | | | | |
| GPU | SoA\_GS\_GPU | 7.343 | 4.825 | 4.626 | 5.082 | 5.24 |
| 68.84352 | 104.7706 | 109.2776 | 99.47226 | 96.4729 |
| AoS\_GS\_GPU | 5.897 | 4.34 | 3.725 | 4.399 | 4.491 |
| 83.1611 | 112.9956 | 131.6513 | 111.4801 | 109.1964 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Image1 | LWS | (8,16) | (16,16) | (32,16) | (64,16) | (32,32) |
| CPU | SoA\_SO\_CPU | 6087.923 | | | | |
| AoS\_SO\_CPU | 6049.634 | | | | |
| GPU | SoA\_SO\_GPU | 17.766 | 16.796 | 17.77 | 27.661 | 28.358 |
| 342.6727 | 362.4627 | 342.5956 | 220.0905 | 214.681 |
| AoS\_SO\_GPU | 17.577 | 16.754 | 17.971 | 26.615 | 25.924 |
| 344.179 | 361.0859 | 336.6331 | 227.3017 | 233.3604 |

**4. 결과 분석**

**1) Locality in memory access on CPU and GPU**

CPU에서 grayscale을 계산할 때는 for loop을 수행하면서 off chip memory에서 r,g,b값을 가져온 후 더해서 intensity를 계산하기 때문에 r,g,b,a값이 연속적으로 위치한 AoS가 SoA보다 spatial locality가 좋다. 위의 실험 결과에서는 0번 image를 사용할 때는 SoA가, 1번 image를 사용할 때는 AoS가 더 빠르다.

CPU에서 sobel operator를 수행할 때는 먼저 input image를 grayscale로 바꾼 후 sobel operator를 수행하도록 하였다. 따라서 수행 후에는 Pixel\_Channels\* AoS\_image\_input (Pixel\_Planes SoA\_image\_input)가 grayscale로 바뀌기 때문에 수행시간을 측정할 때마다 새로 이미지를 읽어들이도록 하였고, 시간 측정은 함수 호출 전후 시간만 측정하여 각각의 수행시간을 모두 더한 후 수행 횟수로 나누었다. 이 때에도 grayscale과 같은 이유로 AoS가 SoA보다 spatial locality가 좋은데, 실제 실험 결과에서는 grayscale과 마찬가지로 0번 image를 사용할 때는 SoA가, 1번 image를 사용할 때는 AoS가 더 빠르다.

위와 같이 0번 image에서 SoA가 더 빠르게 수행된 것은 for loop에서 SoA에서 BYTE 배열인 r,g,b,a plane도 연속적으로 사용되기 때문에 이 plane의 원소들이 캐싱이 되어서 빠르게 수행되었다고 추측할 수 있다.

GPU에서는 work group의 work item들이 동시에 한 plane에 access하기 때문에 SoA가 AoS보다spatial locality가 좋다. local work size (32,16)을 기준으로 GPU에서 grayscale을 수행할 때는 0번 이미지는 약 1.215배, 1번 이미지는 약 1.2418배로 모두 AoS가 더 빠르다.

GPU에서 sobel operator를 수행할 때는 grayscale과 sobel operator를 따로 구현했던 CPU와는 달리 계산 과정에서 각 data의 grayscale로 변환했을 때의 값을 계산해서 weight를 곱한다. 0번 이미지에서는 AoS가, 1번 이미지에서는 SoA가 더 빨랐으나, 둘의 시간 차이가 유의미하다고 보기 어렵다. AoS에서 각 structure의 원소 개수가 4개이기 때문에 structure당 R이 4바이트씩 떨어져 있다. 따라서 한 work item의 R, G, B, A가 캐싱이 되었을 때 다른 work item의 R, G, B, A값이 cache hit가 일어날 가능성이 있기 때문에 AoS가 빠르게 수행되었다고 추측할 수 있다.

**2) work group size**

GPU에서 수행할 때 work group size를 (8,16),(16,16),(32,16),(64,16),(32,32)로 변경시키면서 측정하였다. warp 단위가 32이고, 2차원 work group을 1차원으로 선형화했을 때 work group의 각 row가 image의 width만큼 떨어져 있기 때문에 dim 0을 32의 배수로, work group을 직사각형으로 하면 수행 시간이 짧게 걸릴 것이라고 추측할 수 있다. 실제 실험 결과에서는 grayscale에서 dim1을 16으로 동일하게 하고 dim 0 을 8,16,32,64로 변화시켰을 때 image 0,1 둘 다 dim 0이 32일 때가 가장 수행시간이 적었고, 8에서 가장 수행시간이 오래 걸렸다. (32,16)과 (32,32)를 비교했을 때는 image 0, 1 둘다 (32,16)이 수행시간이 더 적게 걸렸다.

sobel operator에서는 dim 1을 16으로 동일하게 하고 dim 0 을 8,16,32,64로 변화시켰을 때 image 0의 SoA는 (8,16)이, AoS는 (16,16), image 1는 둘 다 (16,16)일 때 수행시간이 가장 적게 걸렸고, image0,1 둘 다 (64,16)일 때 수행시간이 제일 오래 걸렸다. (32,16)과 (32,32)를 비교했을 때는 image 0, 1 둘다 (32,16)이 수행시간이 더 적게 걸렸다.

따라서 dim0이 같을 때는 정사각형 보다 dim1이 작은 직사각형의 work group이 수행시간이 빠르다는 것을 확인할 수 있고, grayscale에서는 (32,16)일 때의 수행시간이 제일 빨랐고, sobel operator에서는 dim0이 warp 단위보다 작은 경우에 (32,16)보다 수행시간이 빠르게 나온 것을 확인하였다.