HW 6 (2021-27764 안지수)

1. Matrix Multiplication with OpenCL

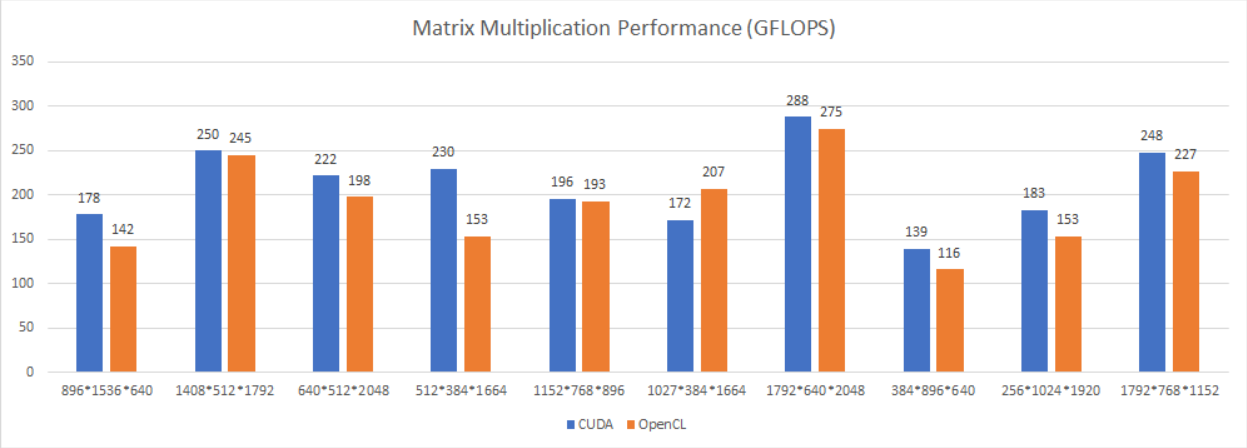
(a) 두가지 포인트로 병렬화를 수행하였다.

(1) 행렬 C의 한 원소를 계산하기 위해 K번 Local memory에 접근하는 부분을 float형 변수를 선언하여 레지스터에서 연산을 수행하고 모든 연산이 끝난 후에 행렬 C의 원소를 업데이트 (Local memory에 접근) 하는 방식으로 변경하였다. 즉, Local Memory의 접근을 최소화 함으로써 성능을 향상시켰다.

(2) Nvidia GPU의 경우 warp(32 thread)단위로 실행되기 때문에 local work size를 32의 배수로 하였고, Turing architecture의 경우 하나의 SM에 4개의 warp scheduler가 존재하여 4개의 warp가 동시에 실행됨으로 local work size를 warp 4개의 배수, 즉 128의 배수로 설정하였다.

(b) OpenCL과 CUDA의 성능 차이를 비교하였다. (조건 local warp size = (2, 64), 단위 GFLOPS)

측정 결과 CUDA가 대체적으로 좋은 성능을 냄을 확인할 수 있었다.



(c) 모두 Valid

```
  Problem size: M = 896, N = 1536, K = 640
  Number of iterations: 1
  Number of warmup iterations: 0
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Calculating...(iter=0)
Calculating done!(iter=0): 0.009852 sec
Validating...
Result: VALID
Reference time: 0.114224 sec
Reference throughput: 15.422356 GFLOPS
Your Avg. time: 0.009852 sec
Your Avg. throughput: 178.815245 GFLOPS
```

```
  Problem size: M = 1408, N = 512, K = 1792
  Number of iterations: 1
  Number of warmup iterations: 0
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Calculating...(iter=0)
Calculating done!(iter=0): 0.010303 sec
Validating...
Result: VALID
Reference time: 0.056584 sec
Reference throughput: 45.660803 GFLOPS
Your Avg. time: 0.010303 sec
Your Avg. throughput: 250.760501 GFLOPS
```

```
  Problem size: M = 640, N = 512, K = 2048
  Number of iterations: 1
  Number of warmup iterations: 0
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Calculating...(iter=0)
Calculating done!(iter=0): 0.006045 sec
Validating...
Result: VALID
Reference time: 0.047040 sec
Reference throughput: 28.532387 GFLOPS
Your Avg. time: 0.006045 sec
Your Avg. throughput: 222.013058 GFLOPS
```

```
  Problem size: M = 512, N = 384, K = 1664
  Number of iterations: 1
  Number of warmup iterations: 0
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Calculating...(iter=0)
Calculating done!(iter=0): 0.002837 sec
Validating...
Result: VALID
Reference time: 0.024392 sec
Reference throughput: 26.824489 GFLOPS
Your Avg. time: 0.002837 sec
Your Avg. throughput: 230.610345 GFLOPS
```

```
  Problem size: M = 1152, N = 768, K = 896
  Number of iterations: 1
  Number of warmup iterations: 0
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Calculating...(iter=0)
Calculating done!(iter=0): 0.008073 sec
Validating...
Result: VALID
Reference time: 0.058530 sec
Reference throughput: 27.087761 GFLOPS
Your Avg. time: 0.008073 sec
Your Avg. throughput: 196.389059 GFLOPS
```

```
  Problem size: M = 1024, N = 384, K = 1664
  Number of iterations: 1
  Number of warmup iterations: 0
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Calculating...(iter=0)
Calculating done!(iter=0): 0.007572 sec
Validating...
Result: VALID
Reference time: 0.087514 sec
Reference throughput: 14.953378 GFLOPS
Your Avg. time: 0.007572 sec
Your Avg. throughput: 172.831400 GFLOPS
```

```
  Problem size: M = 1792, N = 640, K = 2048
  Number of iterations: 1
  Number of warmup iterations: 0
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Calculating...(iter=0)
Calculating done!(iter=0): 0.016304 sec
Validating...
Result: VALID
Reference time: 0.054336 sec
Reference throughput: 86.454504 GFLOPS
Your Avg. time: 0.016304 sec
Your Avg. throughput: 288.132860 GFLOPS
```

```
  Problem size: M = 384, N = 896, K = 640
  Number of iterations: 1
  Number of warmup iterations: 0
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Calculating...(iter=0)
Calculating done!(iter=0): 0.003149 sec
Validating...
Result: VALID
Reference time: 0.106449 sec
Reference throughput: 4.137227 GFLOPS
Your Avg. time: 0.003149 sec
Your Avg. throughput: 139.874253 GFLOPS
```

```
  Problem size: M = 256, N = 1024, K = 1920
  Number of iterations: 1
  Number of warmup iterations: 0
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Calculating...(iter=0)
Calculating done!(iter=0): 0.005482 sec
Validating...
Result: VALID
Reference time: 0.091298 sec
Reference throughput: 11.025749 GFLOPS
Your Avg. time: 0.005482 sec
Your Avg. throughput: 183.610459 GFLOPS
```

```
  Problem size: M = 1792, N = 768, K = 1152
  Number of iterations: 1
  Number of warmup iterations: 0
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Calculating...(iter=0)
Calculating done!(iter=0): 0.012756 sec
Validating...
Result: VALID
Reference time: 0.078289 sec
Reference throughput: 40.502358 GFLOPS
Your Avg. time: 0.012756 sec
Your Avg. throughput: 248.583304 GFLOPS
```

(d) 1194.563423 GFLOPS의 결과를 얻었다.

```
shpc121@a00:~/snu_shpc21/hw6/mat_mul$ make performance
salloc --nodes=1 --ntasks-per-node=1 --cpus-per-task=32 --gres=gpu:1 --partition=shpc mpirun ./main -v -w 1 -n 1 8192 8192 8192
salloc: Pending job allocation 69694
salloc: job 69694 queued and waiting for resources
salloc: job 69694 has been allocated resources
salloc: Granted job allocation 69694
Options:
  Problem size: M = 8192, N = 8192, K = 8192
  Number of iterations: 1
  Number of warmup iterations: 1
  Print matrix: off
  Validation: on
Initializing matrices...
Initializing matrices done!
Initializing...
Initializing done!
Warming up...
Warming up done!: 1.044119 sec
Calculating...(iter=0)
Calculating done!(iter=0): 0.920430 sec
Validating...
Result: VALID
Reference time: 13.041978 sec
Reference throughput: 84.305589 GFLOPS
Your Avg. time: 0.920430 sec
Your Avg. throughput: 1194.563423 GFLOPS
Finalizing...
Finalizing done!
salloc: Relinquishing job allocation 69694
```

2. Double Buffering

Double buffering을 적용하게 되면 성능 향상이 존재하는 것을 확인할 수 있다.

2.98GB 의 vecter addition을 계산한 결과는 다음과 같다.

Single buffer + Non-Blocking : 0.7759 GFLOPS

Single buffer + Blocking: 0.7435 GFLOPS

Double buffer + Non-blocking: 1.0492 GFLOPS

Double buffer + Blocking:0.9852 GFOPS

즉, 실험 결과 Double buffer + Non-blocking을 했을 때 가장 좋은 성능이 나옴을 확인할 수 있었다.

[Single buffering + Non-Blocking]

[Sigle buffering + Blocking]

```
salloc --nodes=2 --ntasks-per-node=1 --cpus-per-task=16 --gres=gpu:1 --partition=shpc mpirun ./main 400000000
salloc: Pending job allocation 70499
salloc: job 70499 queued and waiting for resources
salloc: job 70499 has been allocated resources
salloc: Granted job allocation 70499
Initializing vectors...
Initializing vectors done!
Initializing...
ajs: 48
ajs: 55
Initializing...
ajs: 48
ajs: 55
Initializing done!
Warming up...
Initializing done!
Warming up...
Warming up done!: 1.431028 sec
Calculating...(iter=0)
Warming up done!: 1.734532 sec
Calculating...(iter=0)
Calculating done!(iter=0): 0.985427 sec
Elapsed time using normal I/O: 0.985427 sec
Reference throughput: 0.811830 GFLOPS
Finalizing...
Calculating done!(iter=0): 0.762469 sec
Elapsed time using normal I/O: 0.762469 sec
Reference throughput: 1.049223 GFLOPS
Finalizing...
Finalizing done!
Finalizing done!
CORRECT VEC_ADD!!!
salloc: Relinquishing job allocation 70499
```

[Double buffering + Non-Blocking]

```
salloc --nodes=2 --ntasks-per-node=1 --cpus-per-task=16 --gres=gpu:1 --partition=shpc mpirun ./main 400000000
salloc: Pending job allocation 70514
salloc: job 70514 queued and waiting for resources
salloc: job 70514 has been allocated resources
salloc: Granted job allocation 70514
Initializing vectors...
Initializing vectors done!
Initializing...
Initializing...
Initializing done!
Warming up...
Initializing done!
Warming up...
Warming up done!: 1.413101 sec
Calculating...(iter=0)
Warming up done!: 1.720533 sec
Calculating...(iter=0)
Calculating done!(iter=0): 0.986162 sec
Calculating done!(iter=0): 0.811981 sec
Elapsed time using normal I/O: 0.811981 sec
Reference throughput: 0.985245 GFLOPS
Finalizing...
Elapsed time using normal I/O: 0.986162 sec
Reference throughput: 0.811226 GFLOPS
Finalizing...
Finalizing done!
Finalizing done!
CORRECT VEC_ADD!!!
```

[Double buffering + Blocking]

3. Using Profiler

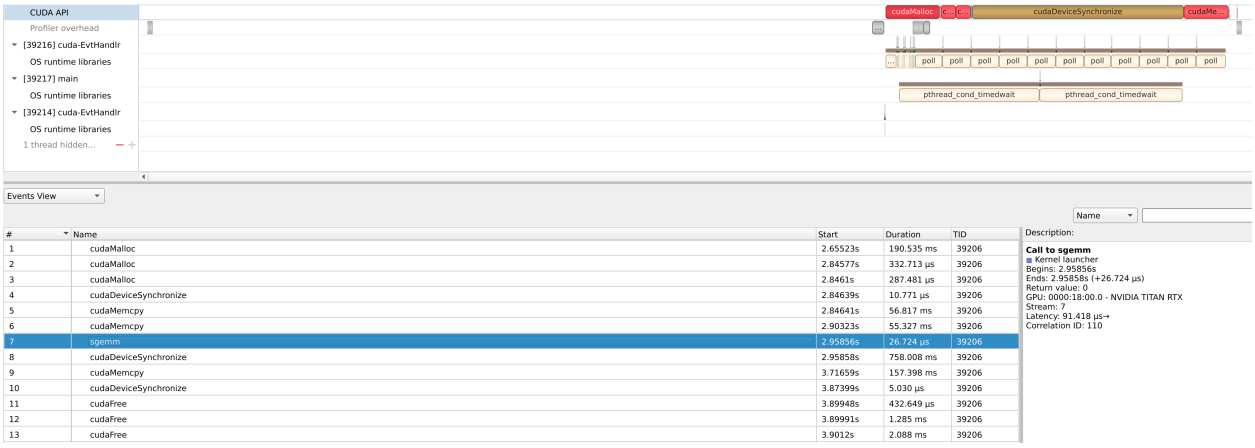
3.2 NVIDIA Nsight Systems

(a) 소요시간 26.724 μs

Begins: 2.95856s

Ends: 2.95858s (+26.724 μs)

(b)



#	Name	Start	Duration	TID
1	cudaMalloc	2.65523s	190.535 ms	39206
2	cudaMalloc	2.84577s	332.713 μs	39206
3	cudaMalloc	2.8461s	287.481 μs	39206
4	cudaDeviceSynchronize	2.84639s	10.771 μs	39206
5	cudaMemcpy	2.84641s	56.817 ms	39206
6	cudaMemcpy	2.90323s	55.327 ms	39206
7	sgemm	2.95856s	26.724 μs	39206
8	cudaDeviceSynchronize	2.95858s	758.008 ms	39206
9	cudaMemcpy	3.71659s	157.398 ms	39206
10	cudaDeviceSynchronize	3.87399s	5.030 μs	39206
11	cudaFree	3.89948s	432.649 μs	39206
12	cudaFree	3.89991s	1.285 ms	39206
13	cudaFree	3.9012s	2.088 ms	39206

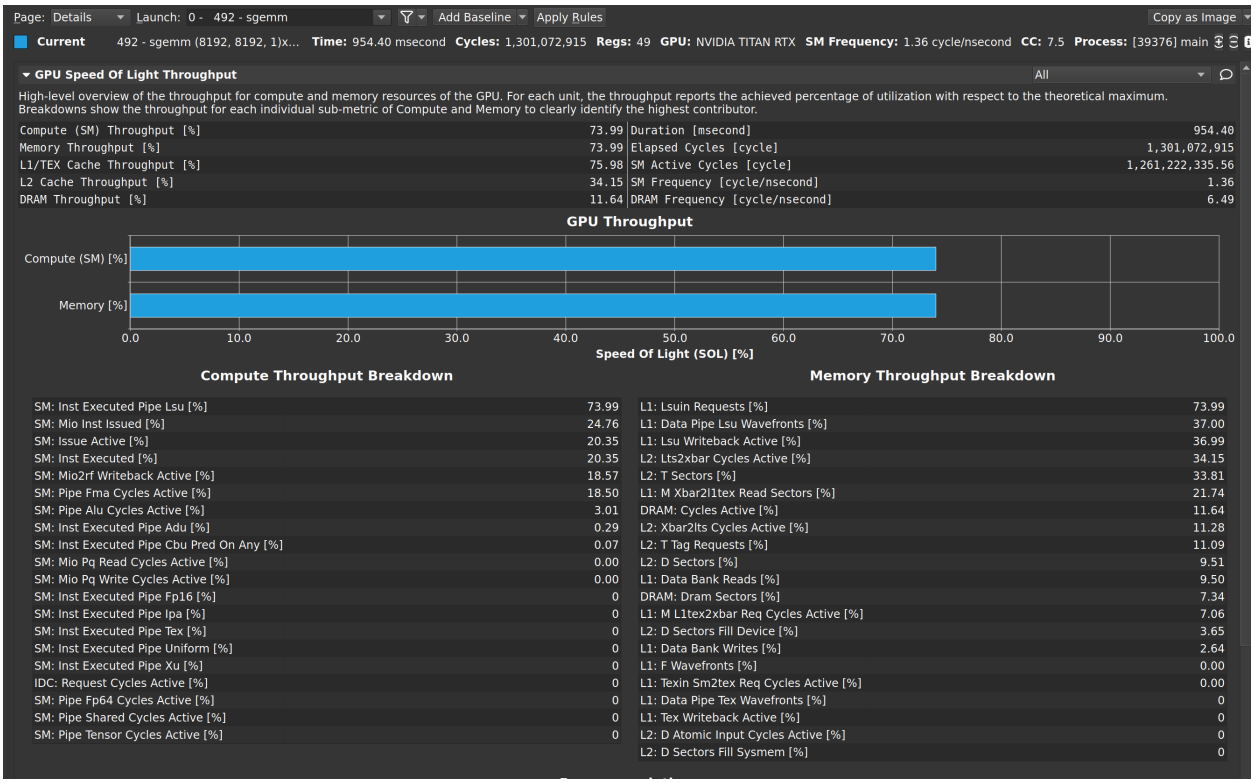
3.3

(a)

SM Throughput 73.99%

Memory Throughput 73.99%

(b)



```
salloc --nodes=2 --ntasks-per-node=1 --cpus-per-task=16 --gres=gpu:1 --partition=shpc mpirun ./main 400000000
salloc: Pending job allocation 70514
salloc: job 70514 queued and waiting for resources
salloc: job 70514 has been allocated resources
salloc: Granted job allocation 70514
Initializing vectors...
Initializing vectors done!
Initializing...
Initializing...
Initializing done!
Warming up...
Initializing done!
Warming up...
Warming up done!: 1.413101 sec
Calculating...(iter=0)
Warming up done!: 1.720533 sec
Calculating...(iter=0)
Calculating done!(iter=0): 0.986162 sec
Calculating done!(iter=0): 0.811981 sec
Elapsed time using normal I/O: 0.811981 sec
Reference throughput: 0.985245 GFLOPS
Finalizing...
Elapsed time using normal I/O: 0.986162 sec
Reference throughput: 0.811226 GFLOPS
Finalizing...
Finalizing done!
Finalizing done!
CORRECT VEC_ADD!!!
```

소감

(a) OpenCL은 CUDA보다 다양한 플랫폼을 제공하기 위해서 조금 더 많은 설정을 해야하고, 비교적 낮은 성능을 가지고 있었다. 하지만 여러 플랫폼을 General 하게 지원할 수 있다는 점이 큰 매력으로 다가왔고, CUDA는 쉽게 코드를 작성할 수 있다는 점이 매력으로 다가왔다. OpenCL과 CUDA 모두 재미있는 프로그래밍 경험이었다.

(b) 과제 퀄리티와 해설지 퀄리티가 너무 좋아 공부하는 데에 많은 도움이 되었다. 만약 이전 수업과제도 받을 수 있다면 받아서 공부해보고 싶다. 그런 점에서 과제가 조금 더 많았으면 더 많이 배울 수 있었을텐데 라는 생각이 든다. (과제와 수업 모두 너무 좋아서 감사한 수업이었다)