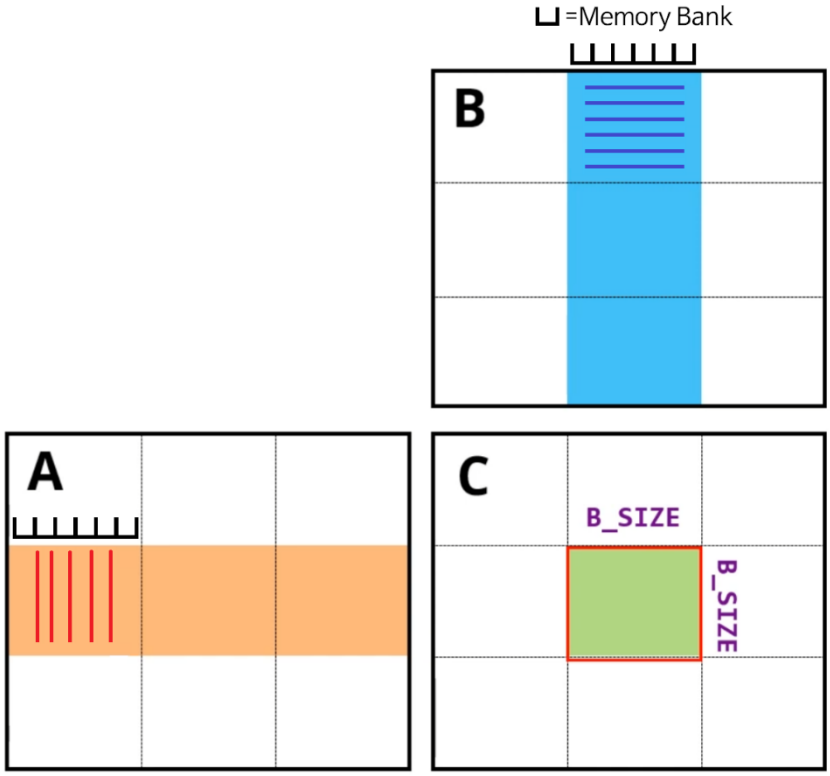
목차

1. 시도한 최적화 기법과 분석

1. 공유 메모리 접근 방식 최적화



[Lab 5-2의 메모리 접근 방식]

GPU의 공유 메모리를 이용한 행렬 곱 알고리즘 과제였던 Lab 5-2에서는 메모리 액세스 패턴에 대한 최적화가 이루어지지 않았다. 특히 Warp 단위로 명령이 실행되는 GPU 상에서, Lab 5-2가 취하였던 공유 메모리 접근 방식은 행렬 B의 공유 메모리에 접근하는 경우에는 각 메모리 뱅크를 스레드가 하나씩 접근하여 메모리 액세스의 병목이 생기지 않는다. 하지만 행렬 A의 공유 메모리에 접근하는 경우 메모리 뱅크의 방향과 동일한 방향으로 각 스레드들의 메모리 액세스가 발생하므로, 성능 하락이 발생한다. 따라서 행렬 A의 공유 메모리에 행렬 A를 전치 행렬(Transposed Matrix) 형태로 저장하고 이를 접근하게 하여 성능을 최적화하였다.

1. 실행 결과

```
Timer Basic : 2198.05930 ms (2198.05930 ms)
// Lab 5-1의 일반 행렬곱 알고리즘 커널의 실행 시간
Timer RR : 1558.53090 ms (1558.53090 ms)
// Lab 5-2과 유사한 공유 메모리 행렬곱 알고리즘 커널의 실행 시간
Timer RC : 1542.68790 ms (1542.68790 ms)
// A 행렬을 전치 행렬 처리한 커널의 실행 시간
```

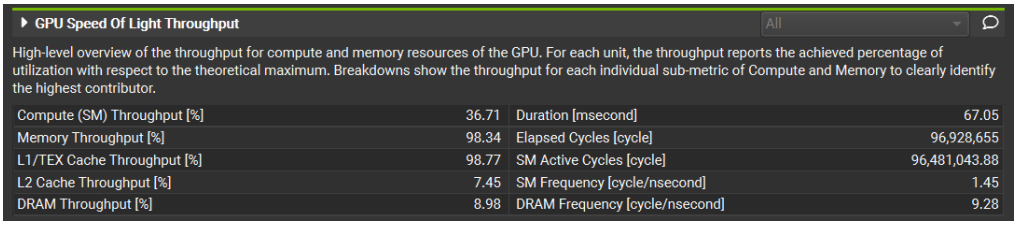
Timer CR : 828.99030 ms (828.99030 ms)
// B 행렬을 전치 행렬 처리한 커널의 실행 시간
Timer CC : 788.86350 ms (788.86350 ms)
// A, B 행렬을 전치 행렬 처리한 커널의 실행 시간

A: 8192 x 8192, B: 8192 x 8192,
Grid: 128x128x1, Block: 16x16x1
CPU: Ryzen 9 3900X, GPU: RTX 3080

위 실행 시간 측정과 함께, NVIDIA Nsight Compute에서 성능 프로파일링을 진행한 결과는 다음과 같다. 본 실험에서는 유의미한 변화가 있었던 프로파일 지표인 "GPU Speed Of Light Throughput"을 수록한다.

2. 프로파일링 결과

1. 전치 없음 (RR)

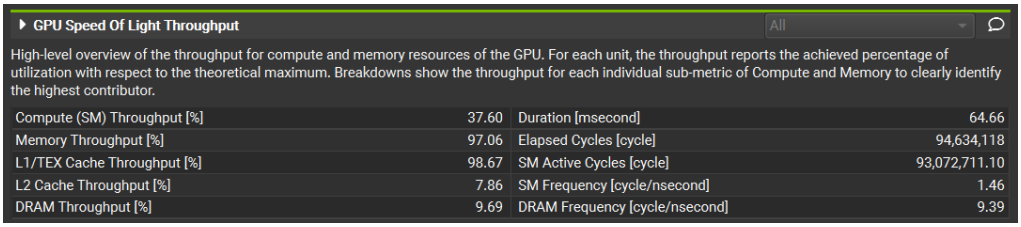


GPU Speed Of Light Throughput — All

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor.

Compute (SM) Throughput [%]	36.71	Duration [msecond]	67.05
Memory Throughput [%]	98.34	Elapsed Cycles [cycle]	96,928,655
L1/TEX Cache Throughput [%]	98.77	SM Active Cycles [cycle]	96,481,043.88
L2 Cache Throughput [%]	7.45	SM Frequency [cycle/nsecond]	1.45
DRAM Throughput [%]	8.98	DRAM Frequency [cycle/nsecond]	9.28

2. B 행렬 전치 (RC)

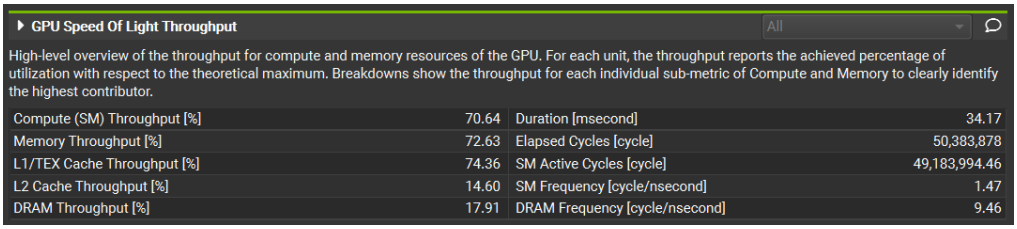


GPU Speed Of Light Throughput — All

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor.

Compute (SM) Throughput [%]	37.60	Duration [msecond]	64.66
Memory Throughput [%]	97.06	Elapsed Cycles [cycle]	94,634,118
L1/TEX Cache Throughput [%]	98.67	SM Active Cycles [cycle]	93,072,711.10
L2 Cache Throughput [%]	7.86	SM Frequency [cycle/nsecond]	1.46
DRAM Throughput [%]	9.69	DRAM Frequency [cycle/nsecond]	9.39

3. A 행렬 전치 (CR)

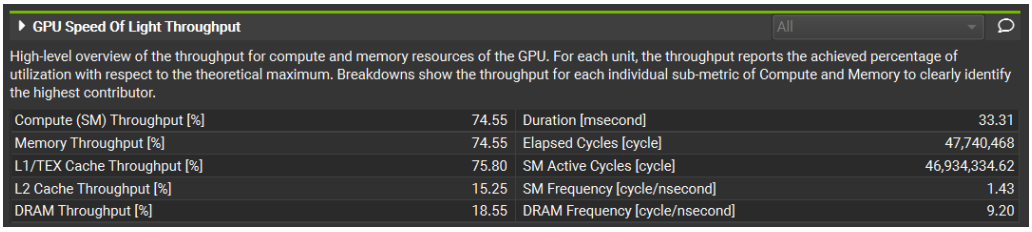


GPU Speed Of Light Throughput — All

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor.

Compute (SM) Throughput [%]	70.64	Duration [msecond]	34.17
Memory Throughput [%]	72.63	Elapsed Cycles [cycle]	50,383,878
L1/TEX Cache Throughput [%]	74.36	SM Active Cycles [cycle]	49,183,994.46
L2 Cache Throughput [%]	14.60	SM Frequency [cycle/nsecond]	1.47
DRAM Throughput [%]	17.91	DRAM Frequency [cycle/nsecond]	9.46

4. A, B 행렬 전치 (CC)



GPU Speed Of Light Throughput — All

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor.

Compute (SM) Throughput [%]	74.55	Duration [msecond]	33.31
Memory Throughput [%]	74.55	Elapsed Cycles [cycle]	47,740,468
L1/TEX Cache Throughput [%]	75.80	SM Active Cycles [cycle]	46,934,334.62
L2 Cache Throughput [%]	15.25	SM Frequency [cycle/nsecond]	1.43
DRAM Throughput [%]	18.55	DRAM Frequency [cycle/nsecond]	9.20

5. 측정 지표

- Compute (SM) Throughput [%]: 이 지표는 GPU의 스트리밍 멀티프로세서(SM)가 계산에 얼마나 많은 시간을 사용하는지를 퍼센트로 표현한다. 이것이 높을수록 GPU는 더 많은 계산을 수행한다.
- Duration [msecond]: CUDA 커널이 실행되는 동안 걸린 시간을 밀리초로 표시한다.
- Memory Throughput [%]: 이 지표는 GPU 메모리 대역폭의 사용률을 나타낸다. 이 값이 높으면 GPU 메모리가 더 많이 사용된다.
- Elapsed Cycles [cycle]: CUDA 커널이 실행되는 동안 걸린 GPU 클럭 사이클 수이다.
- L1/TEX Cache Throughput [%]: L1/TEX 캐시 대역폭의 사용률을 나타낸다. 이 값이 높을수록 이 캐시는 더 많이 사용된다.
- SM Active Cycles [cycle]: SM이 활성 상태로 있던 GPU 클럭 사이클의 총 수이다.
- L2 Cache Throughput [%]: L2 캐시 대역폭의 사용률을 나타낸다. 이 값이 높을수록 이 캐시는 더 많이 사용된다.
- SM Frequency [cycle/nsecond]: SM의 작동 빈도를 나타낸다. 이것은 SM이 초당 얼마나 많은 클럭 사이클을 수행하는지를 나타낸다.
- DRAM Throughput [%]: 이 지표는 DRAM(동적 랜덤 액세스 메모리) 대역폭의 사용률을 나타낸다. 이 값이 높으면 DRAM이 더 많이 사용된다.
- DRAM Frequency [cycle/nsecond]: DRAM의 작동 빈도를 나타낸다. 이것은 DRAM이 초당 얼마나 많은 클럭 사이클을 수행하는지를 나타낸다.

3. 결과 분석

A를 전치한 경우 예상한 대로 큰 폭의 성능 향상이 관측되었다. 특히 Compute Throughput, L1/L2 Cache, DRAM 등의 메모리 관련 Throughput이 큰 폭으로 증가하였고, 실행 시간에 있어서도 거의 2배 수준의 차이가 났다. 한가지 주목할 점은 B 행렬을 전치한 경우인데, 이 경우 성능이 떨어질 것으로 예상했던 것과 달리, 오히려 소폭의 성능 상승이 있었다. 정확한 원인이 파악이 되지는 않지만 GPU의 메모리 구조와 코드에서의 메모리 접근 패턴이 B 행렬을 전치 접근하는

쪽에 유리한 형태를 지니는 것으로 판단된다. (Compute Throughput과 기타 메모리 관련 Throughput이 A 행렬만 전치한 경우보다 A, B를 전치한 경우 약소하게 올라감, 아무것도 전치하지 않은 경우와, B만 전치한 경우를 비교했을 때도 동일한 양상을 보임)

2. 블록 크기 최적화

최적의 성능을 위한 그리드와 블록 크기를 구하기 위해 4x4, 8x8, 16x16, 32x32의 블록 사이즈에 대해서 실행 시간을 측정하고, NVIDIA NSight Compute 프로파일러를 이용하여 성능 지표를 관찰하였다.

1. 실행 결과

[Grid: 2048x2048x1, Block: 4x4x1]
Timer Basic (Lab 5-1): 4505.914600000
Timer RR (전치 없음, Lab 5-2): 5095.705900000
Timer RC (B 전치): 4933.448800000
Timer CR (A 전치): 4951.259800000
Timer CC (A, B 전치): 4989.049500000

[Grid: 512x512x1, Block: 8x8x1]
Timer Basic (Lab 5-1): 1648.140300000
Timer RR (전치 없음, Lab 5-2): 1514.401400000
Timer RC (B 전치): 1490.179100000
Timer CR (A 전치): 1471.410900000
Timer CC (A, B 전치): 1485.477300000

[Grid: 128x128x1, Block: 16x16x1]
Timer Basic (Lab 5-1): 2200.359800000
Timer RR (전치 없음, Lab 5-2): 1565.745800000
Timer RC (B 전치): 1507.470100000
Timer CR (A 전치): 821.570800000
Timer CC (A, B 전치): 789.105300000

[Grid: 32x32x1, Block: 32x32x1]
Timer Basic (Lab 5-1): 4263.692500000
Timer RR (전치 없음, Lab 5-2): 4871.211100000
Timer RC (B 전치): 4703.705600000
Timer CR (A 전치): 1013.889400000

Timer CC (A, B 전치): 905.031300000

A: 8192 x 8192, B: 8192 x 8192

CPU: Ryzen 9 3900X, GPU: RTX 3080

2. 프로파일링 결과

본 실험에서는 주요 프로파일 지표인 "GPU Speed Of Light Throughput"와 Occupancy를 수록한다. 수록된 프로파일 지표들은 A와 B를 전치한의 경우(CC)의 결과들이다.

1. Grid: 2048x2048x1, Block: 4x4x1

GPU Speed Of Light Throughput

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor.

Compute (SM) Throughput [%]	38.76	Duration [second]	5.66
Memory Throughput [%]	38.78	Elapsed Cycles [cycle]	7,833,320,919
L1/TEX Cache Throughput [%]	39.03	SM Active Cycles [cycle]	7,772,677,782.37
L2 Cache Throughput [%]	9.34	SM Frequency [cycle/nsecond]	1.38
DRAM Throughput [%]	14.78	DRAM Frequency [cycle/nsecond]	8.90

Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

Theoretical Occupancy [%]	33.33	Block Limit Registers [block]	48
Theoretical Active Warps per SM [warp]	16	Block Limit Shared Mem [block]	28
Achieved Occupancy [%]	33.33	Block Limit Warps [block]	48
Achieved Active Warps Per SM [warp]	16.00	Block Limit SM [block]	16

2. Grid: 512x512x1, Block: 8x8x1

GPU Speed Of Light Throughput

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor.

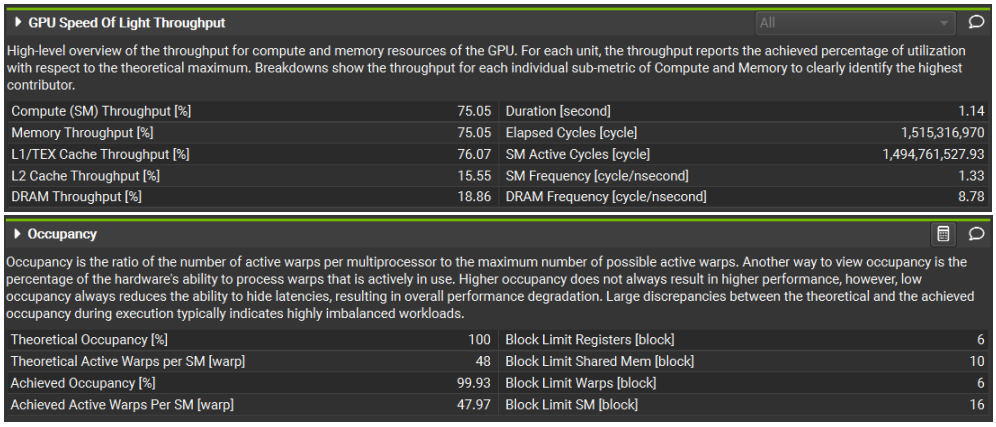
Compute (SM) Throughput [%]	60.31	Duration [second]	1.49
Memory Throughput [%]	60.31	Elapsed Cycles [cycle]	2,095,006,646
L1/TEX Cache Throughput [%]	61.45	SM Active Cycles [cycle]	2,056,151,728.12
L2 Cache Throughput [%]	21.88	SM Frequency [cycle/nsecond]	1.40
DRAM Throughput [%]	26.46	DRAM Frequency [cycle/nsecond]	9.25

Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

Theoretical Occupancy [%]	66.67	Block Limit Registers [block]	24
Theoretical Active Warps per SM [warp]	32	Block Limit Shared Mem [block]	21
Achieved Occupancy [%]	66.73	Block Limit Warps [block]	24
Achieved Active Warps Per SM [warp]	32.03	Block Limit SM [block]	16

3. Grid: 128x128x1, Block: 16x16x1



GPU Speed Of Light Throughput

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor.

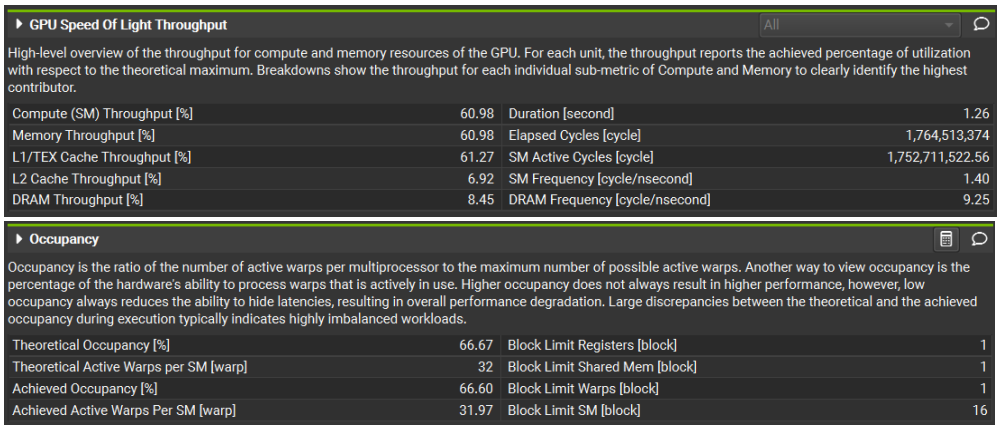
Compute (SM) Throughput [%]	75.05	Duration [second]	1.14
Memory Throughput [%]	75.05	Elapsed Cycles [cycle]	1,515,316,970
L1/TEX Cache Throughput [%]	76.07	SM Active Cycles [cycle]	1,494,761,527.93
L2 Cache Throughput [%]	15.55	SM Frequency [cycle/nsecond]	1.33
DRAM Throughput [%]	18.86	DRAM Frequency [cycle/nsecond]	8.78

Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

Theoretical Occupancy [%]	100	Block Limit Registers [block]	6
Theoretical Active Warps per SM [warp]	48	Block Limit Shared Mem [block]	10
Achieved Occupancy [%]	99.93	Block Limit Warps [block]	6
Achieved Active Warps Per SM [warp]	47.97	Block Limit SM [block]	16

4. Grid: 32x32x1, Block: 32x32x1



GPU Speed Of Light Throughput

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor.

Compute (SM) Throughput [%]	60.98	Duration [second]	1.26
Memory Throughput [%]	60.98	Elapsed Cycles [cycle]	1,764,513,374
L1/TEX Cache Throughput [%]	61.27	SM Active Cycles [cycle]	1,752,711,522.56
L2 Cache Throughput [%]	6.92	SM Frequency [cycle/nsecond]	1.40
DRAM Throughput [%]	8.45	DRAM Frequency [cycle/nsecond]	9.25

Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

Theoretical Occupancy [%]	66.67	Block Limit Registers [block]	1
Theoretical Active Warps per SM [warp]	32	Block Limit Shared Mem [block]	1
Achieved Occupancy [%]	66.60	Block Limit Warps [block]	1
Achieved Active Warps Per SM [warp]	31.97	Block Limit SM [block]	16

5. 측정 지표

- Theoretical Occupancy [%]: 이론적으로 가능한 최대 GPU 스레드 활용도를 나타낸다. GPU에 있는 모든 스레드가 활성화되어 작업을 수행하면 이 값은 100%가 된다.
- Block Limit Registers [block]: 각 블록당 레지스터 수 제한을 나타낸다. 레지스터는 GPU의 SM(스트리밍 멀티프로세서) 내부에 있는 매우 빠른 메모리이다.
- Theoretical Active Warps per SM [warp]: 이론적으로 가능한 SM 당 활성 워프 수를 나타낸다. 워프는 GPU 내의 스레드 그룹이다.
- Block Limit Shared Mem [block]: 각 블록당 공유 메모리 제한을 나타낸다. 공유 메모리는 특정 SM에 있는 모든 스레드가 액세스할 수 있는 메모리이다.
- Achieved Occupancy [%]: 실제로 GPU가 얼마나 많은 스레드를 활성화시켰는지를 나타내는 지표이다. 이 값이 높을수록 GPU가 더 많은 연산을 동시에 수행할 수 있다.
- Block Limit Warps [block]: 각 블록당 워프 수 제한을 나타낸다.
- Achieved Active Warps Per SM [warp]: 실제로 활성화된 SM 당 워프 수를 나타낸다.

- Block Limit SM [block]: SM 당 블록 제한을 나타낸다. 이것은 SM이 동시에 처리할 수 있는 최대 블록 수를 나타낸다. 이 제한은 물리적 하드웨어, 특히 레지스터 및 공유 메모리 사용에 의해 결정된다.

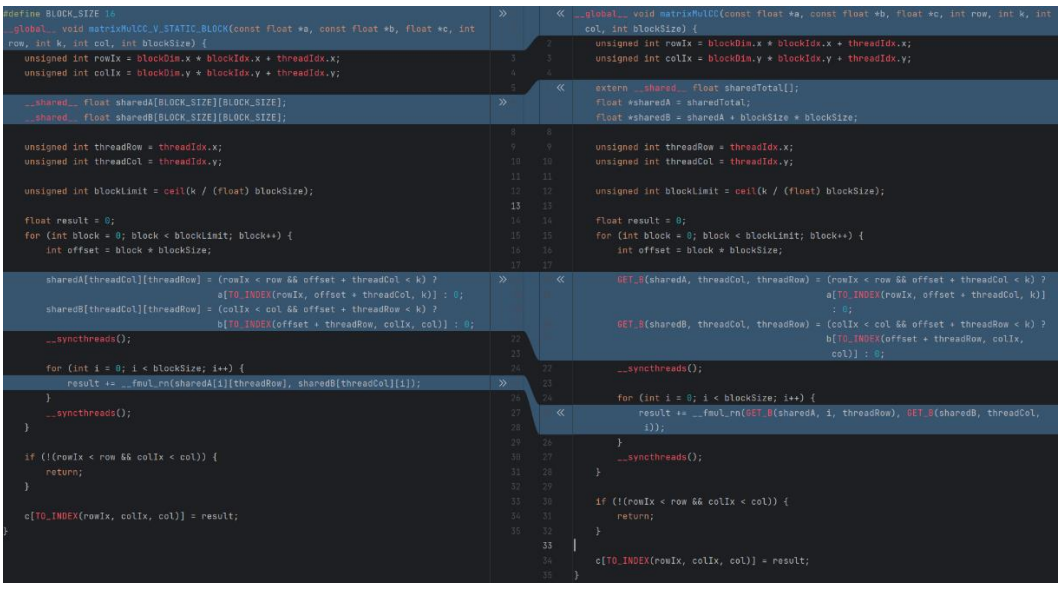
3. 결과 분석

블록 크기가 16x16일 때, Theoretical Occupancy가 100%이 되는 것을 확인할 수 있었으며, Compute Throughput과 기타 메모리 관련 Throughput이 가장 높게 나타났다.

작은 블록 단위(4x4, 8x8)에서는 SM당 활성화시킬 수 있는 Warp 수가 줄어들었고, 전반적으로 블록 크기가 증가할 수록 블록당 레지스터, 공유 메모리, Warp 한도가 감소하였고, 블록 크기가 32x32인 경우에는 블록당 레지스터, 공유 메모리, Warp 한도가 1로 설정되는 것과, SW당 활성화 가능한 Warp 수가 블록 크기가 16x16인 경우보다 줄어든 것을 확인할 수 있었다. 이는 블록 크기가 증가하면, 더 많은 레지스터와 공유 메모리가 필요하고, SM에 할당가능한 블록 수가 줄어들기 때문으로 추정된다.

프로파일링 결과와 함께, 실제 측정된 실행 시간도 A, B 전치 (CC)의 케이스에서 블록 크기가 16x16인 케이스(789MS)가 다른 케이스(4x4, 4989MS)보다 최대 6배 가량 더 빠르게 나타났다.

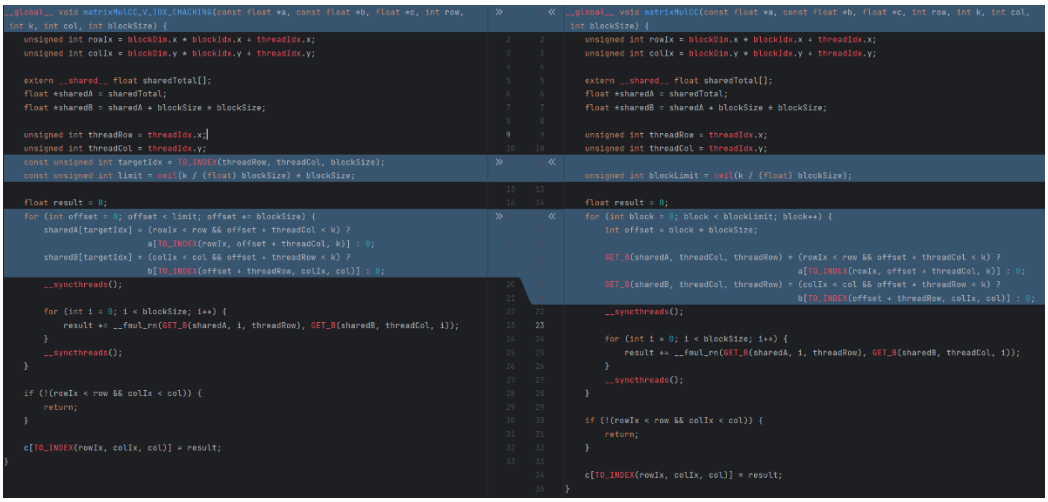
3. 정적 공유 메모리 사용



일반적으로 정적 할당을 사용하면 메모리 주소가 컴파일 타임에 알려지므로, 효율적이고 낮은 런타임 오버헤드를 가지는 메모리 액세스가 가능하다고 알려져 있다. 기존 코드의 경우, 커널 호출시 인자로 공유 메모리의 크기를 넘겨, 공유 메모리 동적 할당을 통해 명령줄 인자에서 들어온 블록 크기만큼 공유 메모리를 할당했는데, 이를 정적 메모리를 사

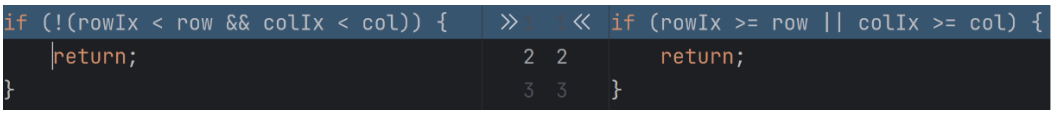
용하도록 바꾸었으나. 두 버전간 유의미한 성능 차이를 확인할 수 없었다.

4. 메모리 색인 계산 최적화



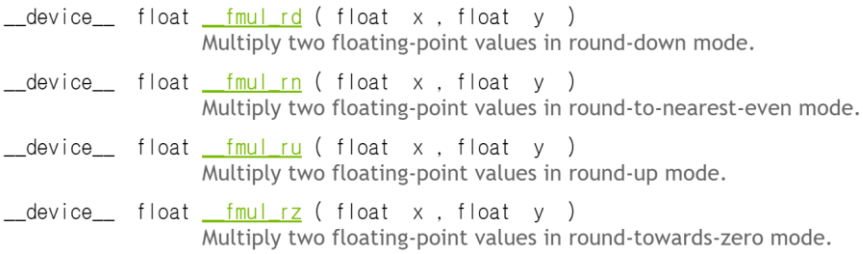
중복된 색인 계산과, 색인 연산간 사용되는 곱 연산의 수를 줄이기 위해서 색인 계산 값을 변수에 저장하여 활용하고, 블록 순회 루프의 루프 변수(block → offest)를 색인 변수화하여, 곱 계산을 줄이는 최적화를 시도했으나 실행 결과는 오히려 미세하게 더 느려진 것을 관찰할 수 있었다. 레지스터 개수 측면에서 색인 변수를 사용하는 것 보다, 계산을 한번 더 진행하는 형태가 더 성능상 이점이 있는 것으로 추정된다.

5. 단락 논리 최적화



예외 조건 검사문에서, 내부 결과를 계산한 뒤 부정연산을 취하는 형태보다. 조건을 뒤집어 앞 조건이 참인 경우, 조건 계산이 바로 완료되게 수정하였다. 의미있는 성능 향상을 보이지는 않았다.

6. 곱 연산 최적화



[단정밀도 곱 내장 함수 (Single Precision Instrinsics)]

행렬 A와 B의 요소를 곱할 때, __fmul_rn (가까운 짝수 반올림) 곱 내장 함수를 사용하고 있었는데, 기본 곱 연산과 더불어 다른 곱 내장 함수 중 반올림과 연산 구현의 차이로 인해 성능 차이를 보일 수 있을 것 같아 모든 종류의 곱 내장 함수를 테스트해 보았지만, 유의미한 성능 차이를 관찰할 수 없었다.

2. 결론

[Case 1]
A: 2048 * 2048, B: 2048 * 2048, Grid: 128x128x1, Block: 16x16x1
Data Transfer Time (Host > Device): 9.994500000
Data Transfer Time (Device > Host): 2.961300000
Timer host (직렬 버전): 55424.781900000
Timer omp (CPU 병렬 버전): 2895.945400000
Timer Basic (Lab 5-1): 40.807300000
Timer RR (전치 없음, Lab 5-2): 29.168000000
Timer RC (B 전치): 27.117200000
Timer CR (A 전치): 16.400000000
Timer CC (A, B 전치): 14.285300000
Timer CC_V_FINAL (최종 버전): 13.963000000

[Case 2]
A: 4096 * 4096, B: 4096 * 4096, Grid: 128x128x1, Block: 16x16x1
Data Transfer Time (Host > Device): 38.112500000
Data Transfer Time (Device > Host): 10.733100000
Timer omp (CPU 병렬 버전): 115792.493400000
Timer Basic (Lab 5-1): 289.746400000
Timer RR (전치 없음, Lab 5-2): 196.751600000
Timer RC (B 전치): 187.545000000
Timer CR (A 전치): 103.301500000
Timer CC (A, B 전치): 100.270500000
Timer CC_V_FINAL (최종 버전): 100.168900000
*. 직렬 버전은 너무 많은 시간이 걸려서 미측정됨

CPU: Ryzen 9 3900X, GPU: RTX 3080

[Case 1] 기준 직렬 처리 버전 대비 4000배 가량, [Case 2] 기준 병렬 처리 버전보다 1000배 이상의 성능 향상을 보여주었다. 또한 GPU상에서의 단순 행렬 곱셈 알고리즘 (Lab 5-1)보다는 3배 가까운 성능 향상, 공유 메모리를 사용했던 Lab 5-2의 알고리즘과 비교하면 2배 정도의 성능 향상을 나타냈다.

3. 소감

이번 과제를 수행하면서 GPU 상에서 어떻게 메모리 레이아웃을 그리고, 활용해야 하는지에 대해서 더 자세하게 이해하게 된 것 같다. 특히 프로파일러를 사용하면서 프로파일러가 측정한 각 항목들에 대해서 공부하고, 왜 이런 성능이 나왔는지에 대해서 분석하면서 GPU가 내부를 알 수 없었던 블랙박스에서 조금씩 윤곽이 보이는 것 같은 느낌이 들었다. 이번 과제에 담을 만큼 진척이 없어서 보고서에서 기술하지는 못했지만, Constant Memory를 잘 활용하거나 (64KB의 용량이 너무 작고, 캐시로 사용하기에는 애매해서 다른 최적화 방안을 찾아보는 쪽으로 최적화를 진행함) Strassen, Winograd 등의 알고리즘을 GPU에서 효율적으로 구현하는 방법도 있을 것 같다는 생각이 들었다. 실제로 Strassen의 알고리즘은 GPU 버전으로 구현된 것들이 (논문과 함께) 존재했다. 기존 Lab 5-1, Lab 5-2 보다 성능을 엄청나게 개선해보고 싶었는데, 생각보다 많이 개선하지 못한 점은 아쉽지만, 나중에 기회가 생긴다면 새롭고 더 나은 방법으로 다시금 도전해보고 싶다는 생각이 들었다.