TA Session 02 – CUDA 프로파일링

Jaehwan Lee
Dept. of Computer Science and Engineering, College of Engineering
Seoul National University

목차

1. Nsight System

Nsight Product Family

- Nsight Systems
  - CUDA 프로그램(system-wide) 프로파일러
  - 프로그램의 전반적인 부분(application-level)을 최적화할 때 사용
- Nsight Compute
  - CUDA 커널 프로파일러
  - 특정 CUDA 커널(kernel-level)을 최적화할 때 사용
- Nsight Graphics
  - CUDA 그래픽 작업 프로파일러

- Nsight Systems로 병목인 지점 및 커널 파악하고,
  이후 Nsight Compute로 커널 최적화 하는 패턴이 일반적

Nsight Systems

- CUDA 프로그램(system-wide) 프로파일러

- 최적화의 가장 첫 단계로 사용
  1. Nsight Systems로 프로그램 전반에 걸친 최적화를 진행하고 병목 지점을 파악
  2. Nsight Compute로 개별 CUDA 커널 최적화

- NVIDIA 공식 웹사이트에서 다운로드 및 프로그램 설치
  - https://developer.nvidia.com/nsight-systems

Nsight Systems 사용법

- 실습서버에서 프로파일링 실행 후 로컬에서 확인
  - 프로파일링 결과가 .nsys-rep 파일로 저장됨

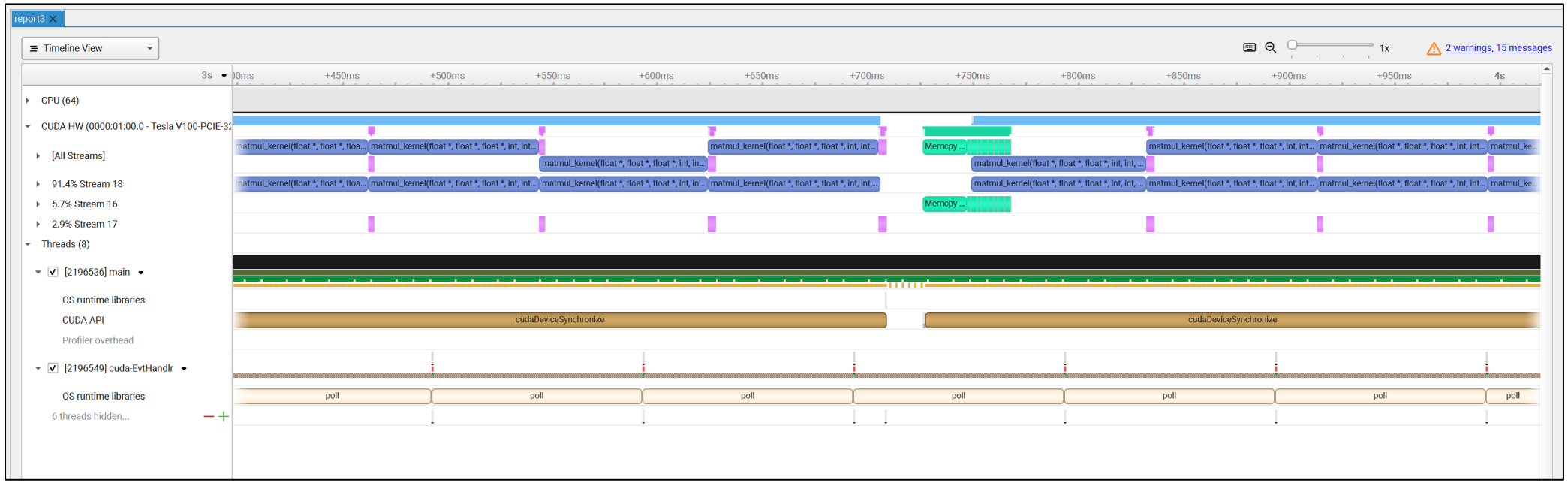
```
$ nsys profile --cudabacktrace=all ./main
```

```
shpcta@login2:~/hw5/matmul_single$ srun --gres=gpu:1 nsys profile --cudabacktrace=all ./main
srun: job 1831610 queued and waiting for resources
srun: job 1831610 has been allocated resources
Options:
  Problem size: M = 8, N = 8, K = 8
  Number of iterations: 1
  Print matrix: off
  Validation: off

Initializing matrices...Done!
Calculating...(iter=0) 0.000048 sec
Avg. time: 0.000048 sec
Avg. throughput: 0.021495 GFLOPS
Generating '/tmp/nsys-report-5203.qdstrm'
[1/1] [========================100%] report3.nsys-rep
Generated:
    /home/s2/shpcta/hw5/matmul_single/report3.nsys-rep
```

Nsight Systems 사용법 (cont'd)

- 로컬에 Nsight Systems를 실행하고 실습서버에서 다운받은 파일을 실행



Nsight Systems CLI - 사용법

```
$ nsys [command_switch] [optional command_switch_options] \
       [application] [optional application_options]
```

- `command_switch`: 프로파일러 기능 지정
  - profile, launch, start, stop, ...
  - 일반적인 경우에는 profile 커맨드로 충분함

- `application`: 실행할 프로그램

- Documentation:
  - https://docs.nvidia.com/nsight-systems/UserGuide/index.html

Nsight Systems CLI - 주요 옵션

- -e, --env-var
  - 프로파일 대상 프로그램 실행 시 환경변수 설정
- -y, --delay
  - 프로그램 실행 이후 일정 시간 이후부터 프로파일링 시작
- -t, --trace
  - 프로파일링 대상 설정
  - cuda, cublas, cudnn, nvtx, mpi 등
- -o, --output
  - 프로파일링 결과 파일 이름
- -f, --force-overwrite
  - 프로파일링 결과 파일이 이미 있으면 덮어쓰기

Nsight Systems CLI - 사용 예시

기본 옵션으로 프로파일링

```
nsys profile ./main ...
```

환경 변수 설정 후 프로파일링 (-e 옵션)

```
nsys profile -e MYVAR=myvar1 ./main ...
```

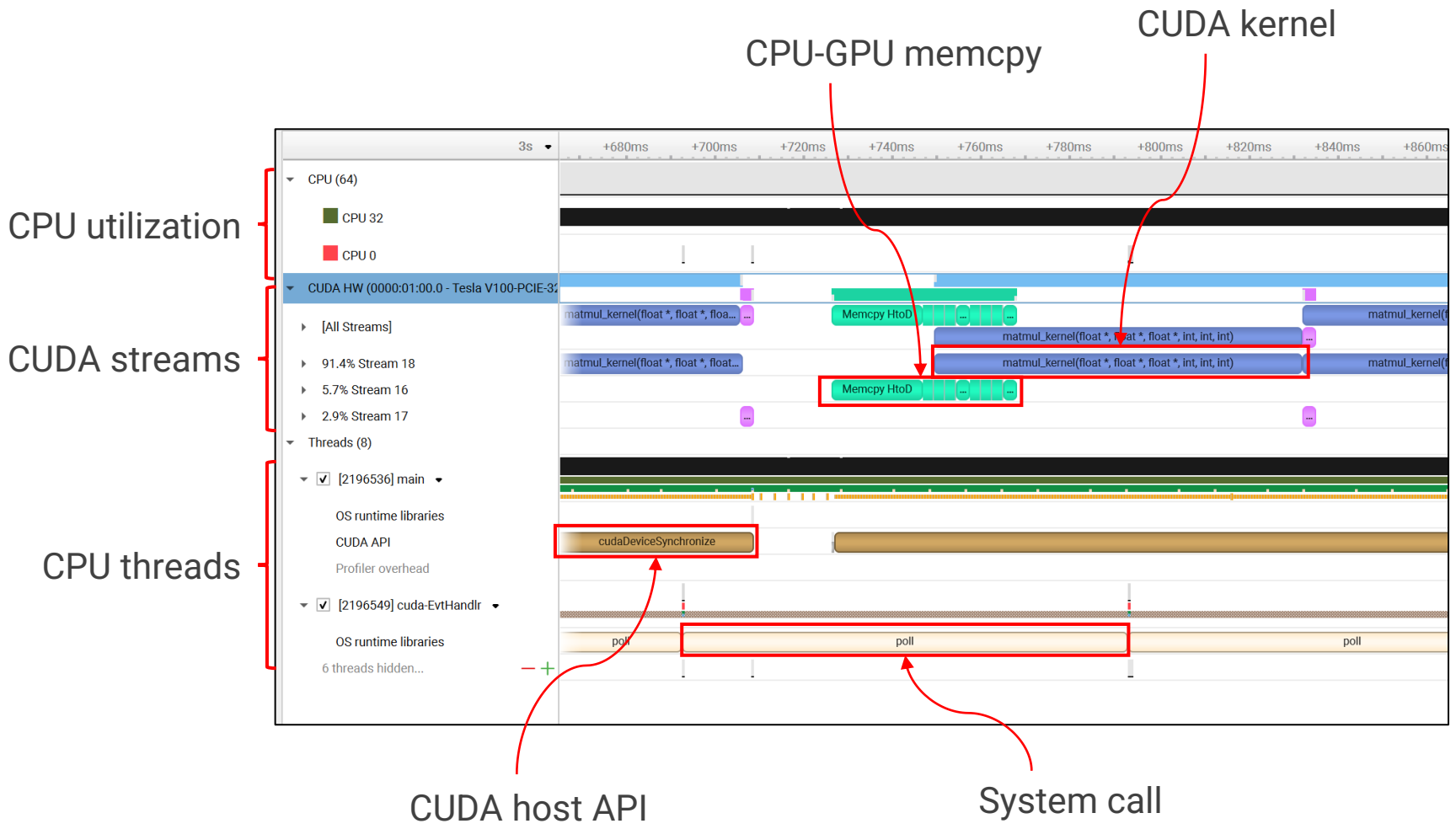
지연된 프로파일링 (--delay 옵션, 프로그램 실행 10초 후 프로파일 시작)

```
nsys profile --delay 10 ./main ...
```

일부 이벤트만 프로파일링 (--trace 옵션)

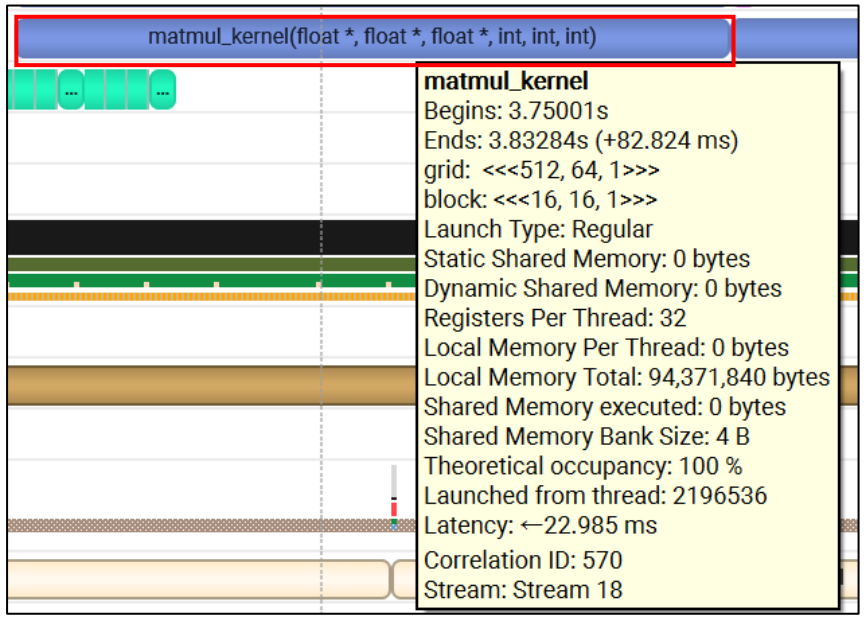
```
nsys profile --trace=cuda,mpi,cudnn,cublas ./main ...
```

Nsight Systems - 프로파일 결과



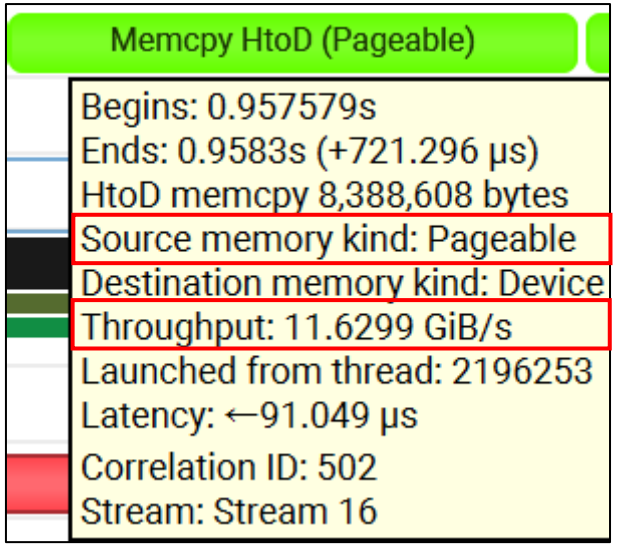
Nsight Systems - 프로파일 결과

- 타임라인의 커널 부분에 마우스를 올려 상세 정보 확인 가능
- 의도한 대로 커널이 실행되었는 지 확인
  - 그리드 및 스레드 블록 크기
  - 공유 메모리 사용량
  - ...



Nsight Systems - 프로파일 결과

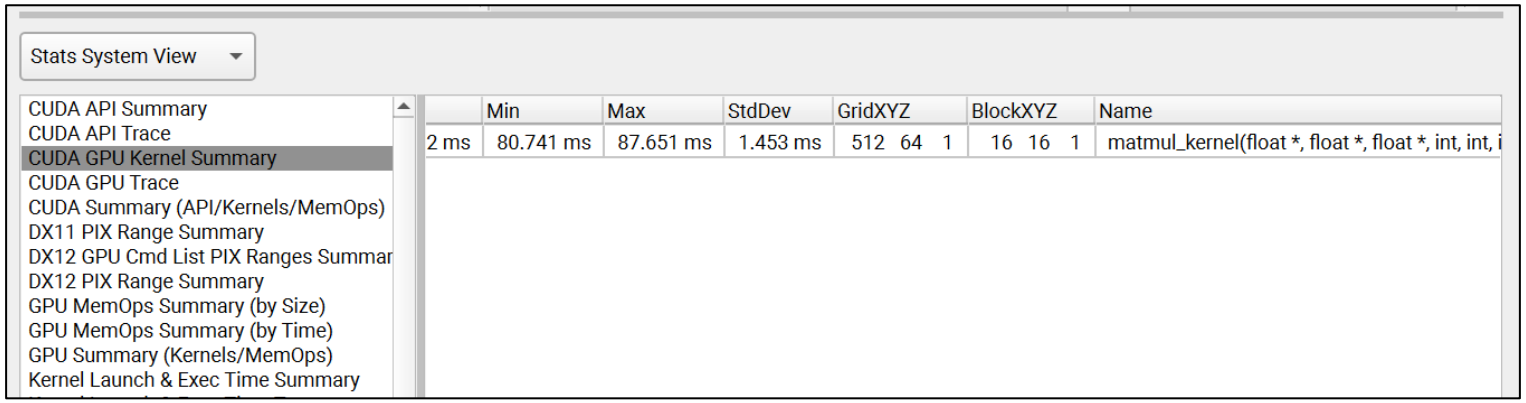
- 타임라인의 커널 부분에 마우스를 올려 상세 정보 확인 가능
- 호스트-디바이스 데이터 전송 대역폭이 정상인지 확인



Nsight Systems - 프로파일 결과

- Stats System View > GPU Kernel Summary 에서 커널별 실행 시간 비중 확인
  - 어떤 커널에서 가장 오랜 시간을 소모하는지 확인에 용이

- 다른 항목들도 확인해 볼 것



Nsight Systems - 프로파일링 구간 지정

- 프로그램의 특정 구간만 프로파일링 하고 싶을 때

- cudaProfilerStart 및 cudaProfilerStop API 사용
  - cuda_profiler_api.h 헤더 파일 include 필요
  - nsys profile 실행 시 `--capture-range cudaProfilerApi` 옵션 설정 필요

프로파일링 구간 지정 예시

```
cudaProfilerStart();
...
cudaProfilerStop();
```

NVTX (NVIDIA Tools Extension)

- Nsight Systems 타임라인에 원하는 이벤트 및 구간을 그리기 위한 라이브러리

- CUDA Toolkit 설치 시 함께 설치됨
  - 소스 코드에 nvToolsExt.h 헤더 파일을 include 해야 함
  - 빌드 시 libnvToolsExt.so 링크해야 함 (`-lnvToolsExt`)

- 자세한 사항은 공식 문서를 찾아보는 것을 추천
  - https://docs.nvidia.com/nvtx/index.html

2. Nsight Compute

Nsight Compute

- CUDA 커널 프로파일러
  - Nsight Systems 프로파일링 결과로 파악한 커널을 더욱 깊게 최적화할 때 사용

- 커널을 반복 실행해 다양한 performance counter 수집
  - 커널 코드에 자동으로 프로파일링 코드 삽입(injection)
  - 커널 실행 직전의 메모리 상태를 저장해두고 반복 실행 시 복원
  - 프로파일링 시간이 길어질 수 있음, 원하는 부분만 프로파일링 하는 것을 추천

Nsight Compute 사용법

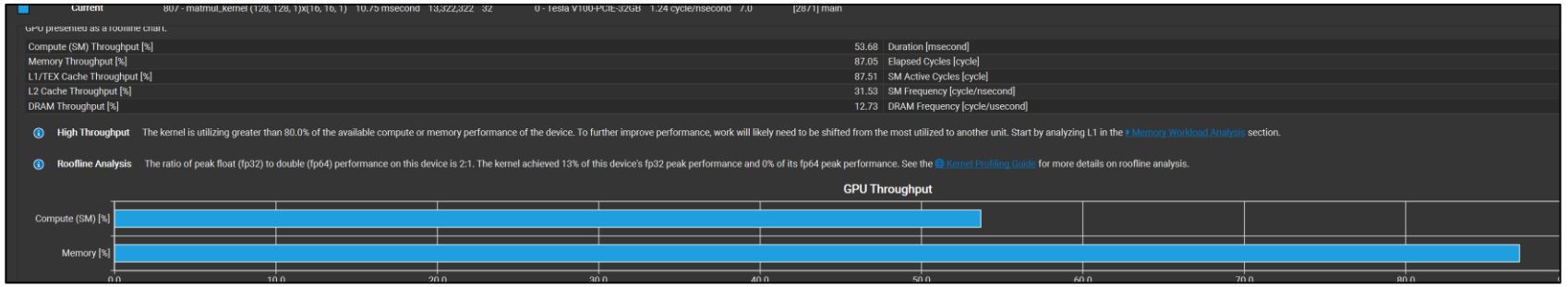
- GPU 서버에서 프로파일링 실행 후 텍스트로 결과 확인

```
$ ncu --set full ./main
```

- GPU 서버에서 프로파일링 실행 후 로컬에서 GUI로 확인
  - ncu CLI 를 이용해 GPU 서버에서 프로파일 데이터 수집
    - 옵션을 통해 프로파일 결과 데이터를 .ncu-rep 파일로 저장

```
$ ncu -o ncu_report --set full ./main
```

  - 프로파일 결과를 로컬로 다운받아 확인



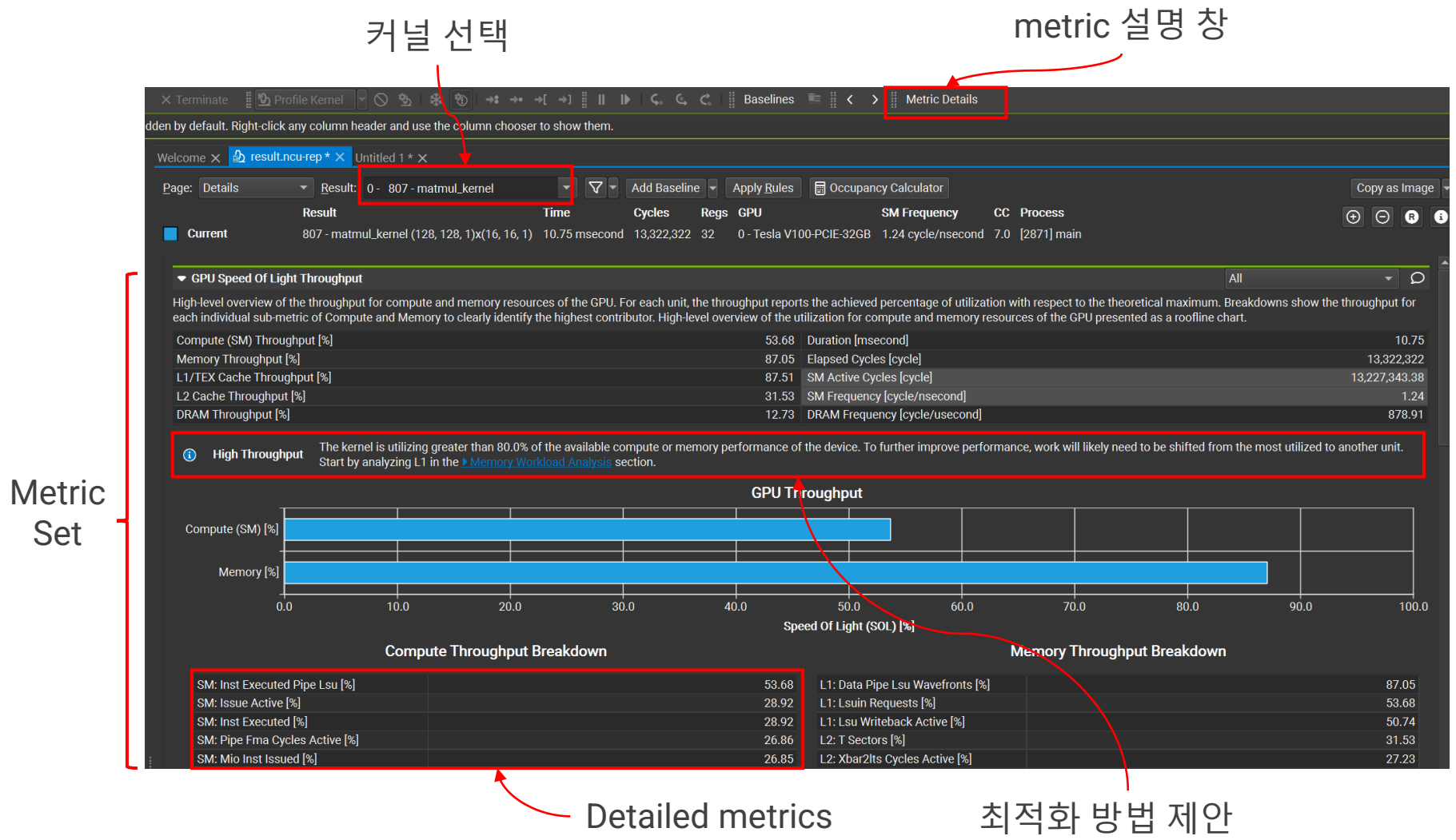
Nsight Compute - CLI 실행 결과 예시

```
shpcta@login2:~/hw5/matmul_single$ TMPDIR=~ srun --gres=gpu:1 ncu --set full ./main
srun: job 1831618 queued and waiting for resources
srun: job 1831618 has been allocated resources
Options:
  Problem size: M = 8, N = 8, K = 8
  Number of iterations: 1
  Print matrix: off
  Validation: off

==PROF== Connected to process 1444749 (/home/s2/shpcta/hw5/matmul_single/main)
==PROF== Profiling "matmul_kernel" - 0: 0%....50%....100% - 31 passes
==PROF== Profiling "matmul_kernel" - 1: 0%....50%....100% - 31 passes
==PROF== Profiling "matmul_kernel" - 2: 0%....50%....100% - 31 passes
Initializing matrices...Done!
Calculating...(iter=0) ==PROF== Profiling "matmul_kernel" - 3: 0%....50%....100% - 31 passes
==PROF== Disconnected from process 1444749
0.343731 sec
Avg. time: 0.343731 sec
Avg. throughput: 0.000003 GFLOPS
[1444749] main@127.0.0.1
  matmul_kernel(float *, float *, float *, int, int, int), 2025-Nov-17 22:45:14, Context 1, Stream 7
    Section: GPU Speed Of Light Throughput
    ---------------------------------------------------------------------- --------------- ------------------------------
    DRAM Frequency                                          cycle/nsecond                           6.28
    SM Frequency                                            cycle/nsecond                           1.32
    Elapsed Cycles                                                  cycle                          5,118
    Memory [%]                                                          %                           0.94
    DRAM Throughput                                                     %                           0.03
    Duration                                                       usecond                          3.87
    L1/TEX Cache Throughput                                             %                           1.57
    L2 Cache Throughput                                                 %                           0.82
    SM Active Cycles                                                cycle                      3,048.60
    Compute (SM) [%]                                                    %                           0.94
    ---------------------------------------------------------------------- --------------- ------------------------------
```

권한 관련 오류 발생 시
TMPDIR 환경변수를 제공할 것!

Nsight Compute - GUI 예시



Nsight Compute - Metric Sets

- GPU Speed of Light Throughput
  - 전반적인 GPU 자원 사용 효율 관련 metric 모음
- Compute Workload Analysis
  - SM 들의 각 연산 유닛 사용량 관련 metric 모음
- Memory Workload Analysis
  - GPU 메모리 계층 구조의 각 부분 사용량 관련 metric 모음
- Scheduler Statistics
- Warp State Statistics
- Instruction Statistics
- ...

Nsight Compute - Metric Sets (cont.)

- ncu --list-sets 명령으로 어떤 metric 모음이 프로파일링 가능한지 확인 가능
- --set 혹은 --metric 옵션으로 원하는 metric 들만 수집하도록 지정 가능
  - 프로파일링 시간에 문제가 없으면 full set 사용 (`$ ncu --set full ...`)

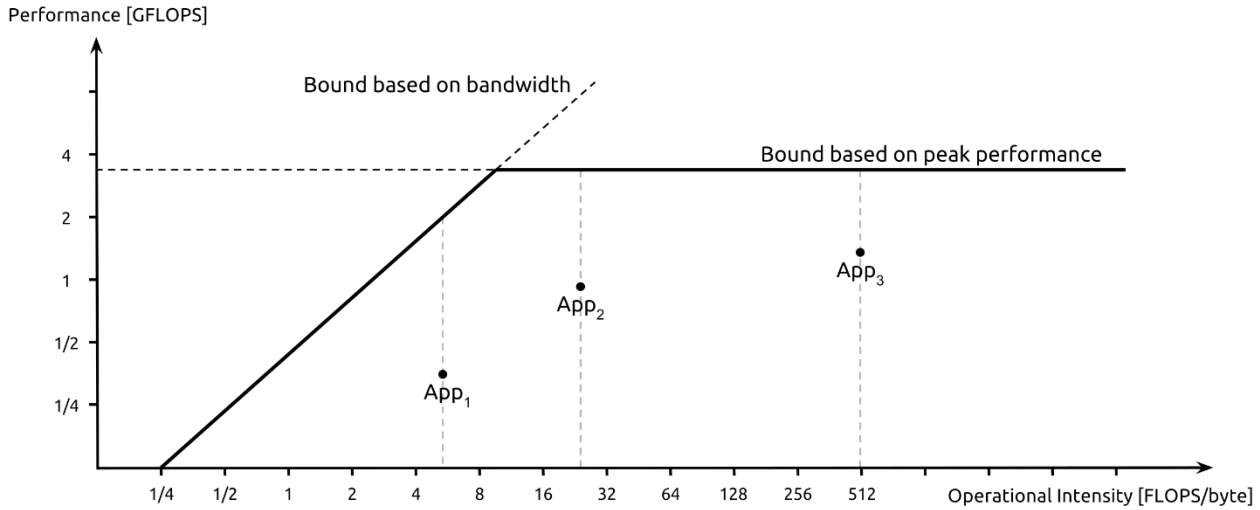
```
$ ncu -list-sets
---------- ------------------------------------------------------------------------------- ------- ----------------
Identifier Sections                                                                        Enabled Estimated Metrics
---------- ------------------------------------------------------------------------------- ------- ----------------
basic      LaunchStats, Occupancy, SpeedOfLight, WorkloadDistribution                      yes     144
detailed   ComputeWorkloadAnalysis, LaunchStats, MemoryWorkloadAnalysis, MemoryWorkloa     no      459
           dAnalysis_Chart, Occupancy, SourceCounters, SpeedOfLight, SpeedOfLight_Roof
           lineChart, WorkloadDistribution
full       ComputeWorkloadAnalysis, InstructionStats, LaunchStats, MemoryWorkloadAnaly     no      613
           sis, MemoryWorkloadAnalysis_Chart, MemoryWorkloadAnalysis_Tables, NumaAffin
           ity, Nvlink_Tables, Nvlink_Topology, Occupancy, PmSampling, SchedulerStats,
            SourceCounters, SpeedOfLight, SpeedOfLight_RooflineChart, WarpStateStats,
           WorkloadDistribution
nvlink     Nvlink, Nvlink_Tables, Nvlink_Topology                                          no      52
pmsampling PmSampling, PmSampling_WarpStates                                               no      72
roofline   SpeedOfLight, SpeedOfLight_HierarchicalDoubleRooflineChart, SpeedOfLight_Hi     no      241
           erarchicalHalfRooflineChart, SpeedOfLight_HierarchicalSingleRooflineChart,
           SpeedOfLight_HierarchicalTensorRooflineChart, SpeedOfLight_RooflineChart, W
           orkloadDistribution
```

GPU SOL Throughput

- 전반적인 GPU 자원 사용 효율 관련 metric 모음
  - Compute (SM) throughput (%)
  - Memory throughput (%)
  - L1/TEX cache throughput (%)
  - L2 cache throughput (%)
  - DRAM throughput (%)
- Throughput은 엄밀한 성능 metric이 아님, 최적화 방향을 결정하는 데에 참고용으로만 사용
  - Breakdown chart에 표시된 다양한 metric의 최댓값을 throughput 으로 표시함
  - 예) Memory throughput 이 예상보다 너무 낮을 경우 메모리 접근 최적화 진행

Roofline Model

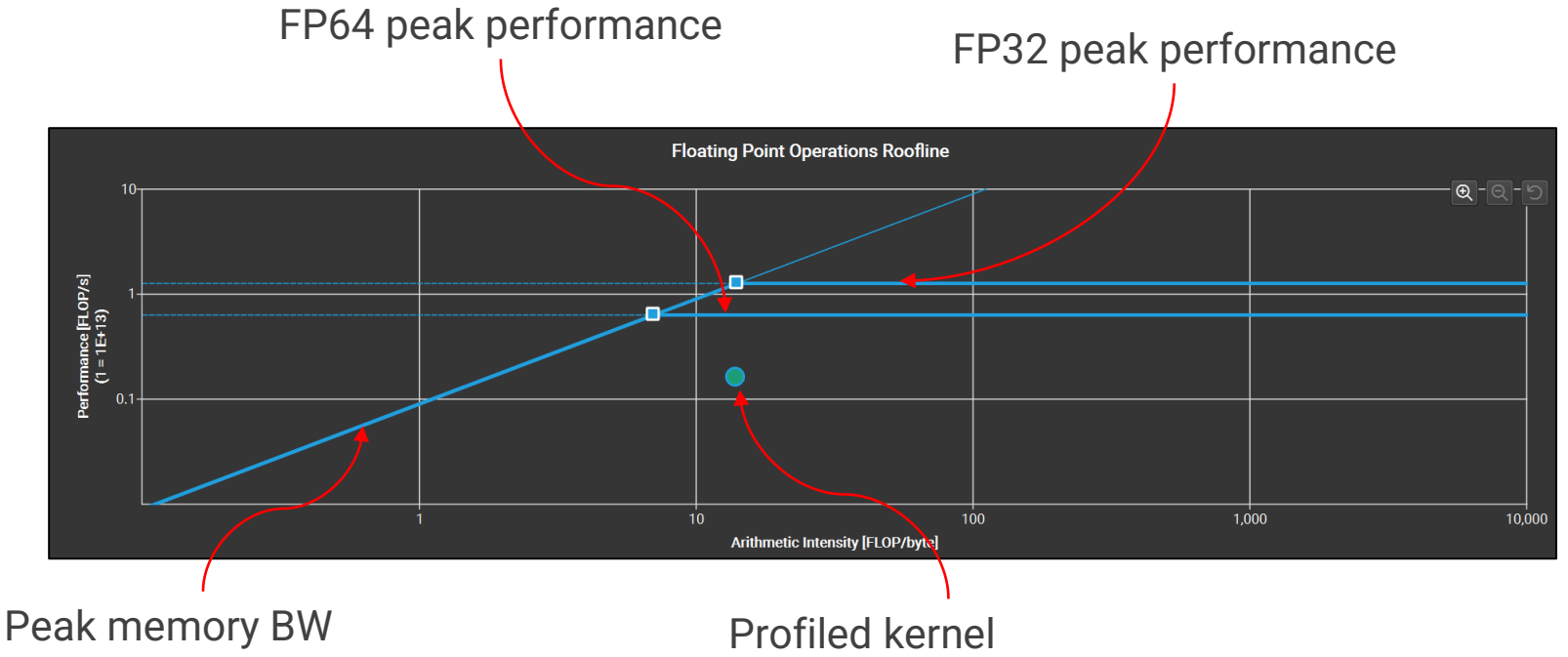
- 커널의 최적화 정도 및 최적화 방향을 파악하기 위한 성능 모델 (Performance Model)
  - [가로축] Operational Intensity: 커널의 메모리 접근 1byte 당 실수 연산 횟수 (FLOP/byte). Arithmetic Intensity라고도 함
  - [세로축] Performance: 커널이 달성한 실수 연산 성능 (FLOP/s)
  - Roofline: 장치의 메모리 대역폭 및 이론상 최대 실수 연산 성능을 그린 것



출처: Wikipedia Roofline Model, https://en.wikipedia.org/wiki/Roofline_model

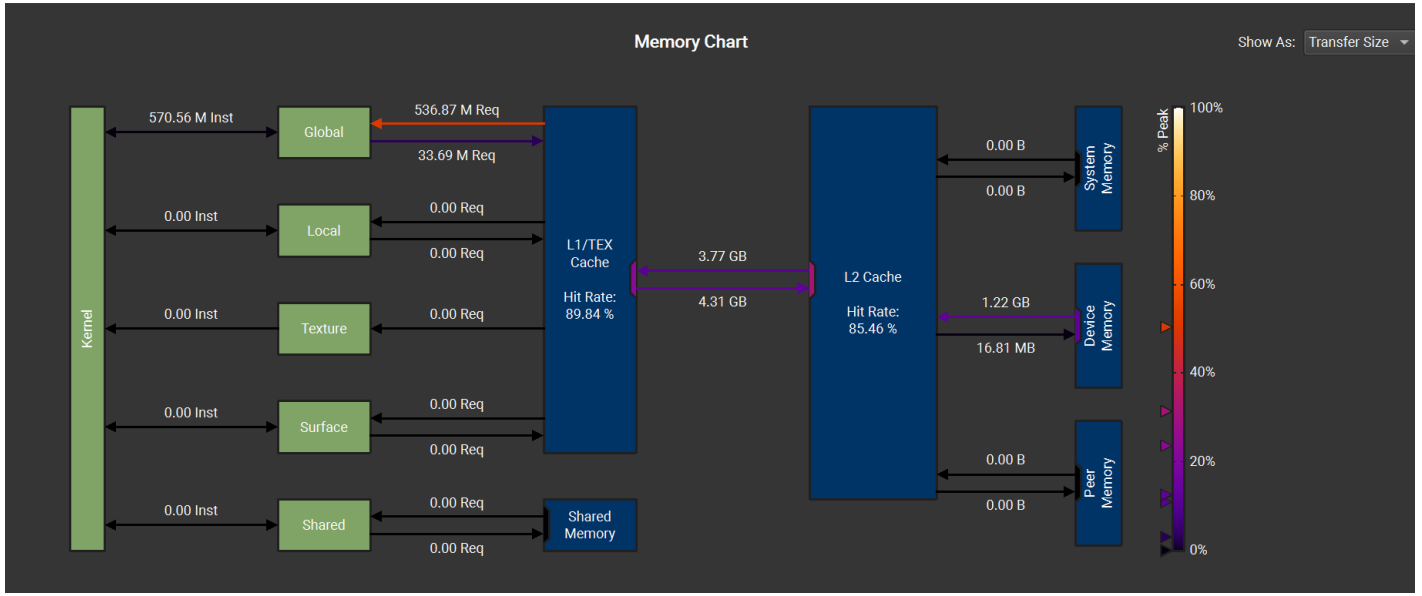
Roofline Model (cont.)

- 커널 프로파일 결과로 루프라인 모델을 제공
- GPU의 이론상 최고 성능에 얼마나 근접한지 확인하고, 최적화 진행 여부 및 방향을 결정



Memory Chart

- 커널 프로파일 결과로 GPU 메모리 계층 구조의 각 구성요소 간 통신량을 보여주는 memory chart 제공
- 공유 메모리를 사용한 최적화를 진행할 때 유용함
  - 의도한대로 커널이 잘 작동하는지 확인



Occupancy

- (# of active warps) / (# of warps supported by SM)

- Active warp: Warp-level resources(레지스터 등)를 할당받은 warp

- SM 하나에 동시에 배정되는 스레드 블록 개수는 HW 자원에 영향을 받음
  - Maximum warps / thread blocks per SM
  - Registers / shared memory per SM

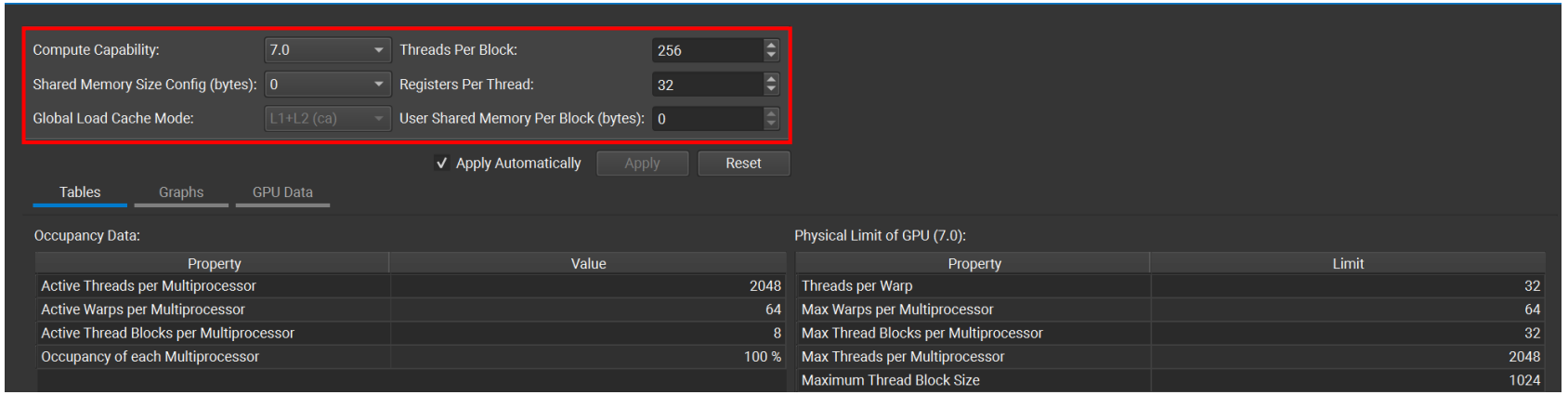
Occupancy (cont.)

- Occupancy가 낮다는 것은 SM이 실행할 수 있는 워프 후보의 개수가 적다는 의미임
  - Hardware Context Switching을 통한 latency hiding이 어려움

- Occupancy를 높이려면?
  - 커널 코드를 최적화해 레지스터와 공유 메모리를 적게 사용하도록 스레드 블록 크기를 튜닝

Nsight Compute - Occupancy Calculator

- 커널을 튜닝해서 어느 정도의 occupancy를 달성 가능할지 계산해주는 도구
  - 스레드 블록 크기
  - 스레드별 레지스터 사용량
  - 스레드 블록별 공유 메모리 사용량
- GPU 아키텍처, 공유 메모리 크기 또한 지정 가능



Nsight Compute - Kernel Filtering

- 프로파일링 시간 단축을 위해 특정 커널만 프로파일링
- 커널 이름 지정
  - -k, --kernel-name 옵션 사용
  - 이름이 일치하는 특정 커널들만 프로파일링
  - regex 사용 가능

커널 이름 지정을 통해 특정 커널만 프로파일링하는 예시

```
$ ncu -k matmul ...
$ ncu -k regex:mat ...
```

- NVTX 구간 지정
  - --nvtx-include 및 --nvtx-exclude 옵션 사용
  - 여러 구간을 ","로 구분해 지정 가능

NVTX 구간 지정을 통해 특정 커널만 프로파일링하는 예시

```
$ ncu --nvtx --nvtx-include "Range B, Range C"--nvtx-exclude "Range A" ...
```

Nsight Compute 를 이용한 최적화 예시

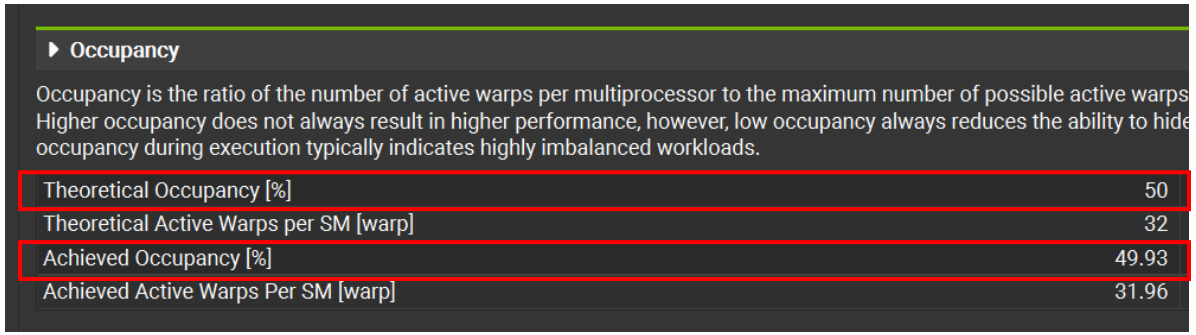
- 가장 기본적인 행렬곱 커널
- 4096 x 4096 x 4096 기준 208.24 GFLOPS
- 문제점은?

```
static __global__ void matmul_kernel(float *A, float *B, float *C, int M, int N,
                                      int K) {
  int i = blockDim.x * blockIdx.x + threadIdx.x;
  int j = blockDim.y * blockIdx.y + threadIdx.y;
  if (i >= M || j >= N) return;
  float sum = 0.0;
  for (int k = 0; k < K; ++k) sum += A[i * K + k] * B[k * N + j];
  C[i * N + j] = sum;
}
```

```
int bsize = 32;
dim3 blockDim(bsize, bsize);
dim3 gridDim((M + bsize - 1) / bsize, (N + bsize - 1) / bsize);
matmul_kernel<<<gridDim, blockDim>>>(A_gpu, B_gpu, C_gpu, M, N, K);
```

Nsight Compute 를 이용한 최적화 예시 (cont.)

- 프로파일링 결과 커널이 너무 낮은 occupancy를 보임



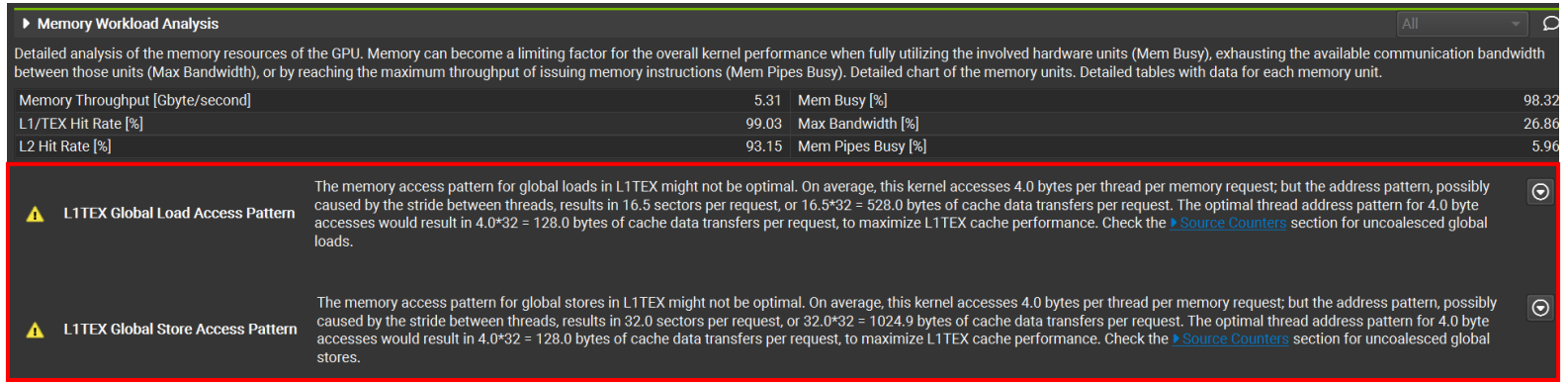
⇒ 스레드 블록 크기 튜닝을 통해 occupancy를 100%로 향상

```
int bsize = 16;
dim3 blockDim(bsize, bsize);
dim3 gridDim((M + bsize - 1) / bsize, (N + bsize - 1) / bsize);
matmul_kernel<<<gridDim, blockDim>>>(A_gpu, B_gpu, C_gpu, M, N, K);
```

- 208.24 GFLOPS → 389.29 GFLOPS (x1.87 speedup)

Nsight Compute 를 이용한 최적화 예시 (cont.)

- 추가 프로파일링 결과, 메모리 접근 패턴에 문제가 있음을 확인
  - Nsight Compute에서 제공하는 최적화 제안 참고



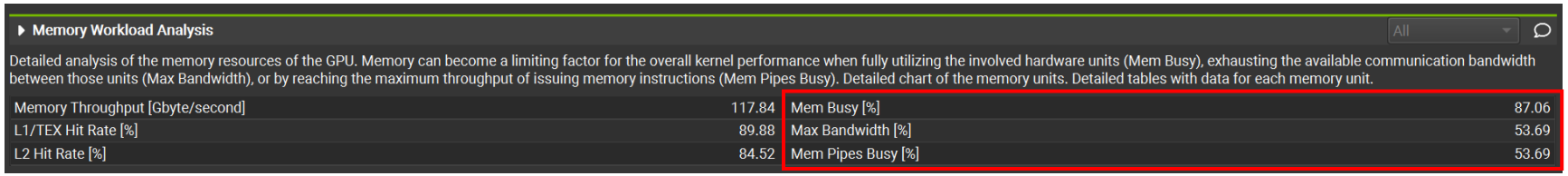
⇒ Memory coalescing 최적화 진행

```
int bsize = 16;
dim3 blockDim(bsize, bsize);
dim3 gridDim((N + bsize - 1) / bsize, (M + bsize - 1) / bsize);
matmul_kernel<<<gridDim, blockDim>>>(A_gpu, B_gpu, C_gpu, M, N, K);
```

- 389.29 GFLOPS → 1563.95 GFLOPS (x4.02 speedup)

Nsight Compute 를 이용한 최적화 예시 (cont.)

- 추가 프로파일링 결과, 메모리 접근에 개선 여지가 더 있음을 확인



Memory Workload Analysis

All

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.

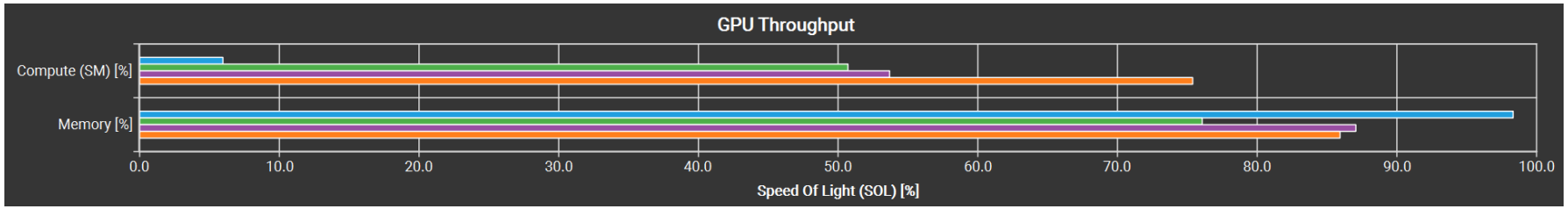
Memory Throughput [Gbyte/second]	117.84	Mem Busy [%]	87.06
L1/TEX Hit Rate [%]	89.88	Max Bandwidth [%]	53.69
L2 Hit Rate [%]	84.52	Mem Pipes Busy [%]	53.69

- ⇒ 공유 메모리를 사용한 최적화 진행

- 1563.95 GFLOPS → 3080.98 GFLOPS (x1.97 speedup)

Nsight Compute 를 이용한 최적화 예시 (cont.)

- 최적화 단계에 따른 GPU Speed of Light metric 변화



- 최적화 단계에 따른 Roofline model 변화