ALCF DevSession, February 23rd, 2022

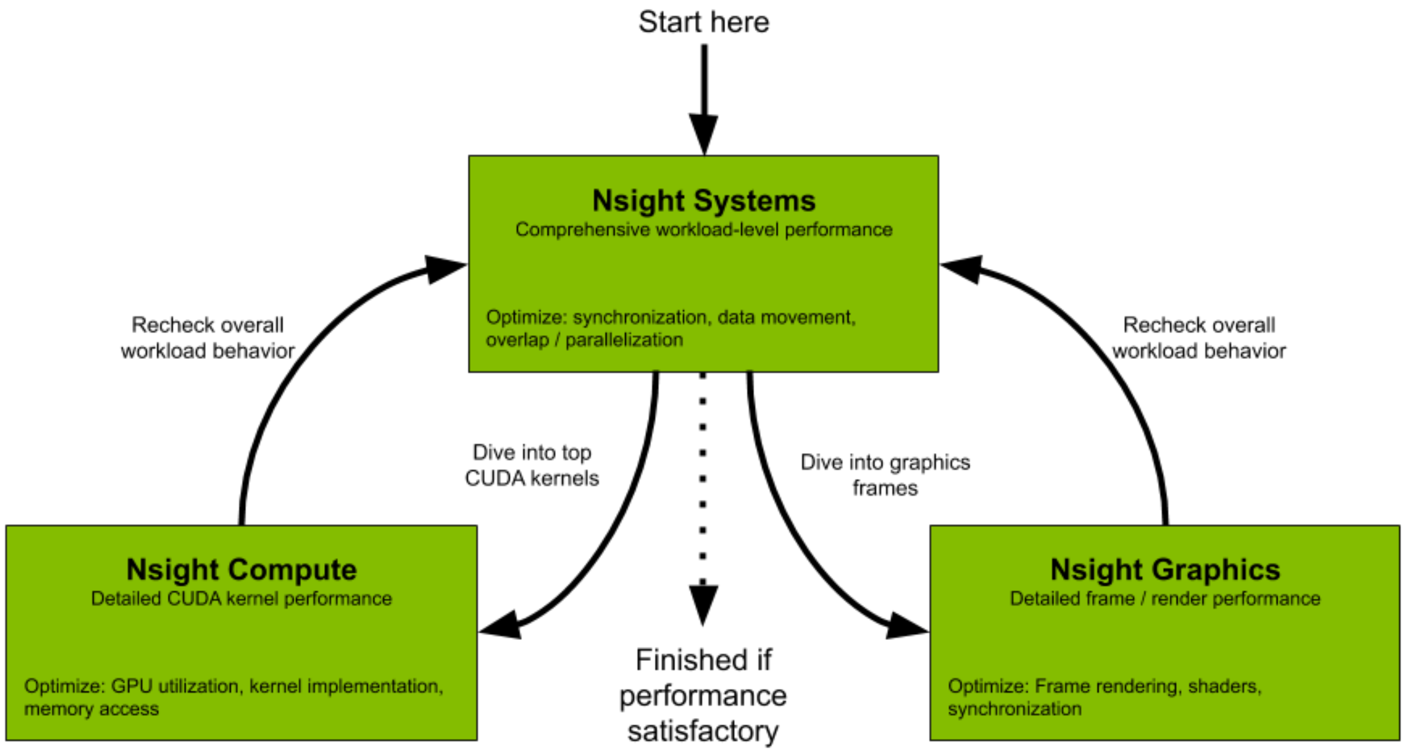
Performance Analysis using Nsight Systems/Compute Tools

JaeHyuk Kwack (jkwack@anl.gov)
ALCF Performance Engineering Group

Argonne NATIONAL LABORATORY

NVIDIA Tools overview

- Nsight Systems
  - —A system-wide visualization of an application performance
  - —To optimize bottlenecks to scale efficiently across CPUs and GPUs on ThetaGPU
- Nsight Compute
  - —An interactive kernel profiler for applications
  - —Providing detailed performance metrics and API debugging via a user interface and command line tool
  - —Providing customizable and data-driven user interface and metric collection that can be extended with analysis scripts for post-processing results
- Nsight Graphics
  - —A stand-alone tool for graphics applications



Credit: NVIDIA (https://developer.nvidia.com/tools-overview)

Step-by-step guide on ThetaGPU (1/2)

- Common part on ThetaGPU
  - Build your application for ThetaGPU
  - Submit your job script to ThetaGPU or start an interactive job mode on ThetaGPU as follows:

```
$ module load cobalt/cobalt-gpu
$ qsub -I -n 1 -t 30 -q full-node -A {your_project}
```

- Nsight Systems
  - Run your application with Nsight Systems as follows:

```
$ nsys profile -o {output_filename} --stats=true ./{your_application}
```

- Nsight Compute
  - Run your application with Nsight Compute

```
$ ncu --set detailed -k {kernel_name} -o {output_filename} ./{your_application}
Or,
$ ncu --set detailed -k regex:"kernel1|kernel2" -o {output_filename}
./{your_application}
```

  - Remark: W/o -o option, Nsight Compute provides performance data as a standard output

https://alcf.anl.gov/support-center/theta-gpu-nodes/nvidia-nsight

Step-by-step guide on ThetaGPU (2/2)

- Post-processing the profiled data
  - Post-processing via CLI

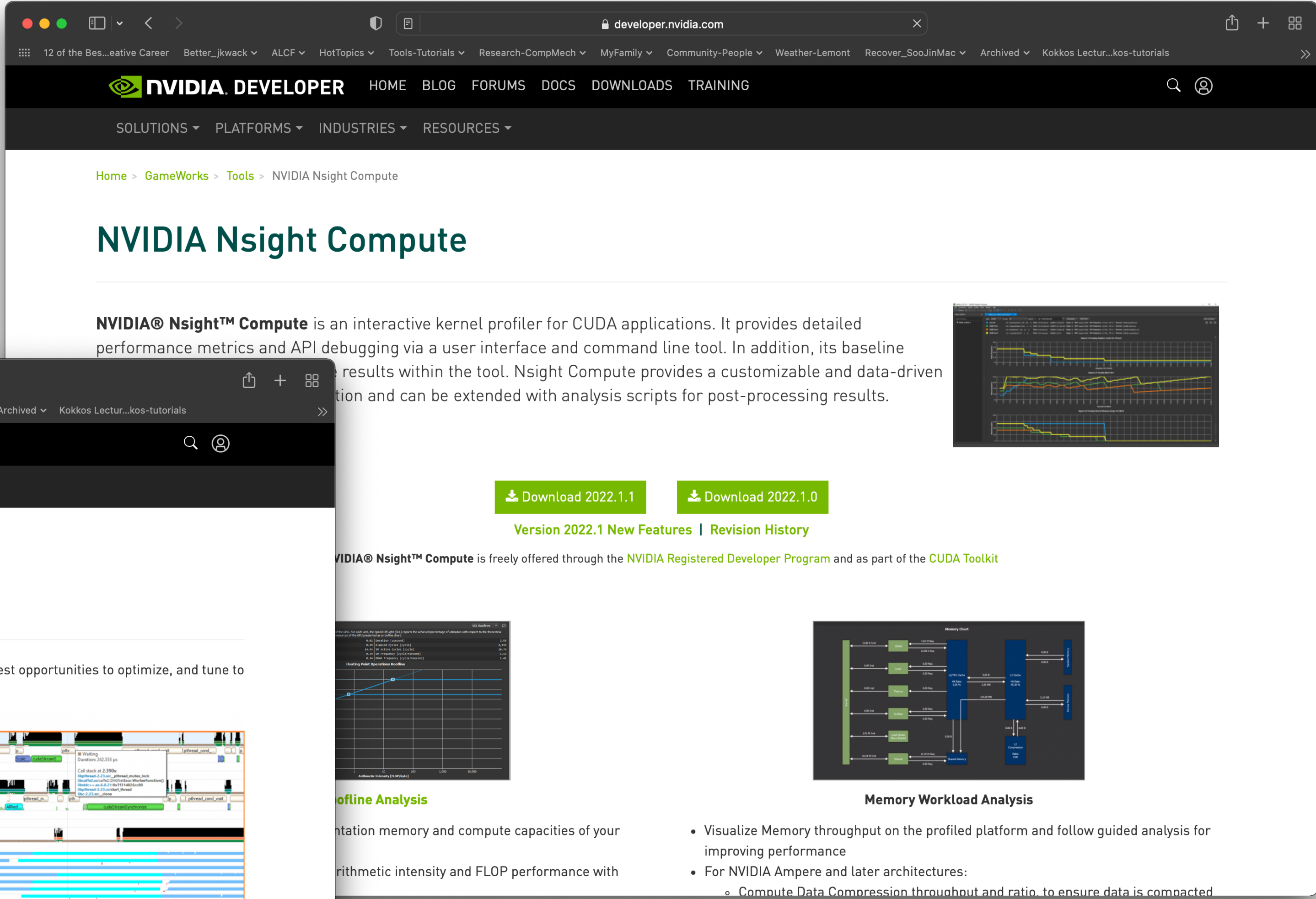
```
$ nsys stats {output_filename}.qdrep
$ ncu -i {output_filename}.ncu-rep
```

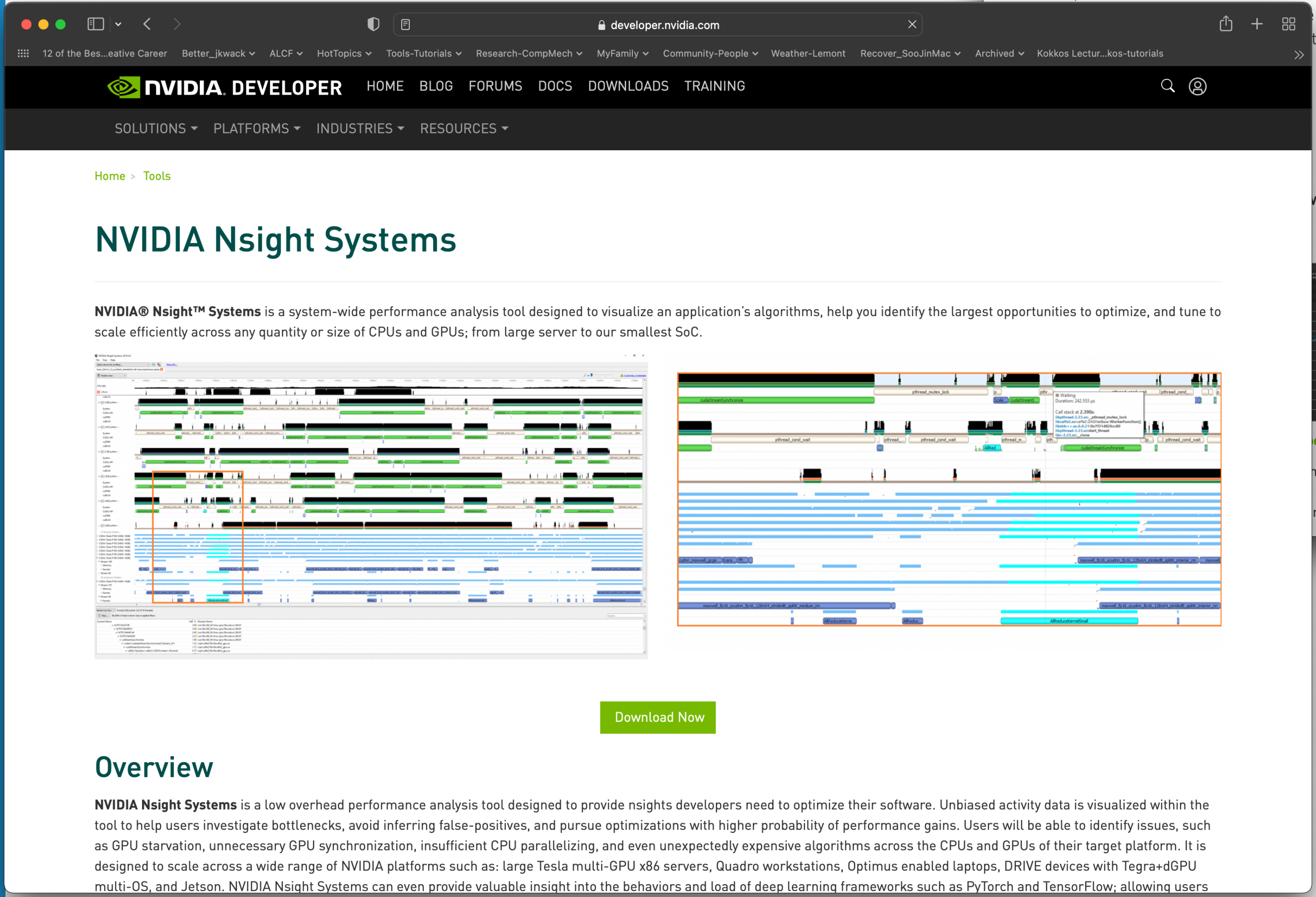
  - Post-processing on your local system via GUI
    - Install NVIDIA Nsight Systems and NVIDIA Nsight Compute after downloading both of them from the NVIDIA Developer Zone.
    - Download nsys output files (i.e., ending with .qdrep and . sqlite) to your local system, and then open them with NVIDIA Nsight Systems on your local system.
    - Download ncu output files (i.e., ending with .ncu-rep) to your local system, and then open them with NVIDIA Nsight Compute on your local system.
  - More options for performance analysis with Nsight Systems and Nsight Compute

```
$ nsys --help
$ ncu --help
```


https://alcf.anl.gov/support-center/theta-gpu-nodes/nvidia-nsight

NVIDIA Developer Zone





https://developer.nvidia.com/nsight-systems
https://developer.nvidia.com/nsight-compute

Argonne NATIONAL LABORATORY

Argonne NATIONAL LABORATORY

Case 1: Stream benchmark

Stream Benchmark

- BabelStream: stream benchmark implemented in multiple programming models
  - https://github.com/UoB-HPC/BabelStream
  - ***CUDA***, OpenMP, SYCL, HIP, OpenCL, Kokkos, RAJA, and so on
- Five kernels to measure memory bandwidth for a given array (size: N)

kernal	source	FLOP	Bytes	Arith. Intensity	Instruction
copy	c[i] = a[i];	0	16N	0	
mul	b[i] = scalar * c[i];	N	16N	0.063	Mul
add	c[i] = a[i] + b[i];	N	24N	0.042	ADD
triad	a[i] = b[i] + scalar * c[i]	2N	24N	0.083	FMA
dot	tb_sum[local_i] += a[i] * b[i];	2N	32N	0.063	FMA

```
template <typename T>
__global__ void add_kernel(const T * a, const T * b, T * c)
{
  const int i = blockDim.x * blockIdx.x + threadIdx.x;
  c[i] = a[i] + b[i];
}
```

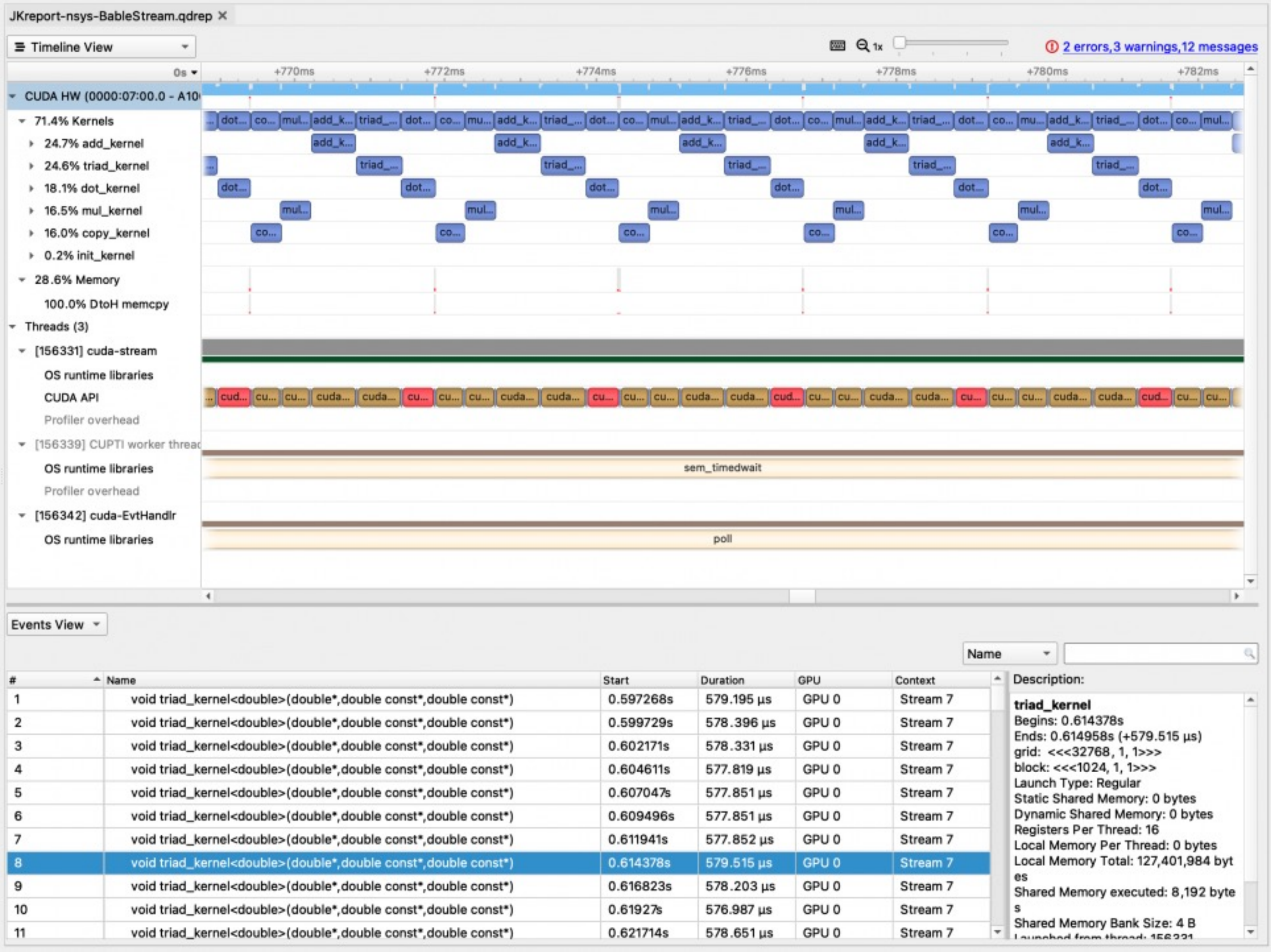
```
template <typename T>
__global__ void triad_kernel(T * a, const T * b, const T * c)
{
  const T scalar = startScalar;
  const int i = blockDim.x * blockIdx.x + threadIdx.x;
  a[i] = b[i] + scalar * c[i];
}
```

Nsight Systems with Stream benchmark

```
$ nsys profile -o JKreport-nsys-BableStream --stats=true ./cuda-stream
…
Generating CUDA API Statistics...
CUDA API Statistics (nanoseconds)
Time(%)      Total Time       Calls          Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------  --------------------------------------------------------------------------------
   44.8       280504347           4      70126086.8         1050249       276881346  cudaMalloc
   31.4       196878210         401        490968.1          381542          600948  cudaDeviceSynchronize
   22.4       140280462         103       1361946.2          436597        32339232  cudaMemcpy
    1.0         6263864           4       1565966.0         1236542         1884610  cudaFree
    0.4         2729558         501          5448.2            4970           36269  cudaLaunchKernel

Generating CUDA Kernel Statistics...
CUDA Kernel Statistics (nanoseconds)
Time(%)      Total Time   Instances          Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------  ------------------------------------------------------------------
   24.7        58518170         100        585181.7          580347          594395  void add_kernel<double>(double const*, double const*, double*)
   24.6        58312184         100        583121.8          576987          595067  void triad_kernel<double>(double*, double const*, double const*)
   18.1        42942748         100        429427.5          419548          438333  void dot_kernel<double>(double const*, double const*, double*, int)
   16.5        39062588         100        390625.9          388733          392125  void mul_kernel<double>(double*, double const*)
   16.0        37980930         100        379809.3          376541          392925  void copy_kernel<double>(double const*, double*)
    0.2          521628           1        521628.0          521628          521628  void init_kernel<double>(double*, double*, double*, double, double, double)
…
```

Nsight Systems with Stream benchmark

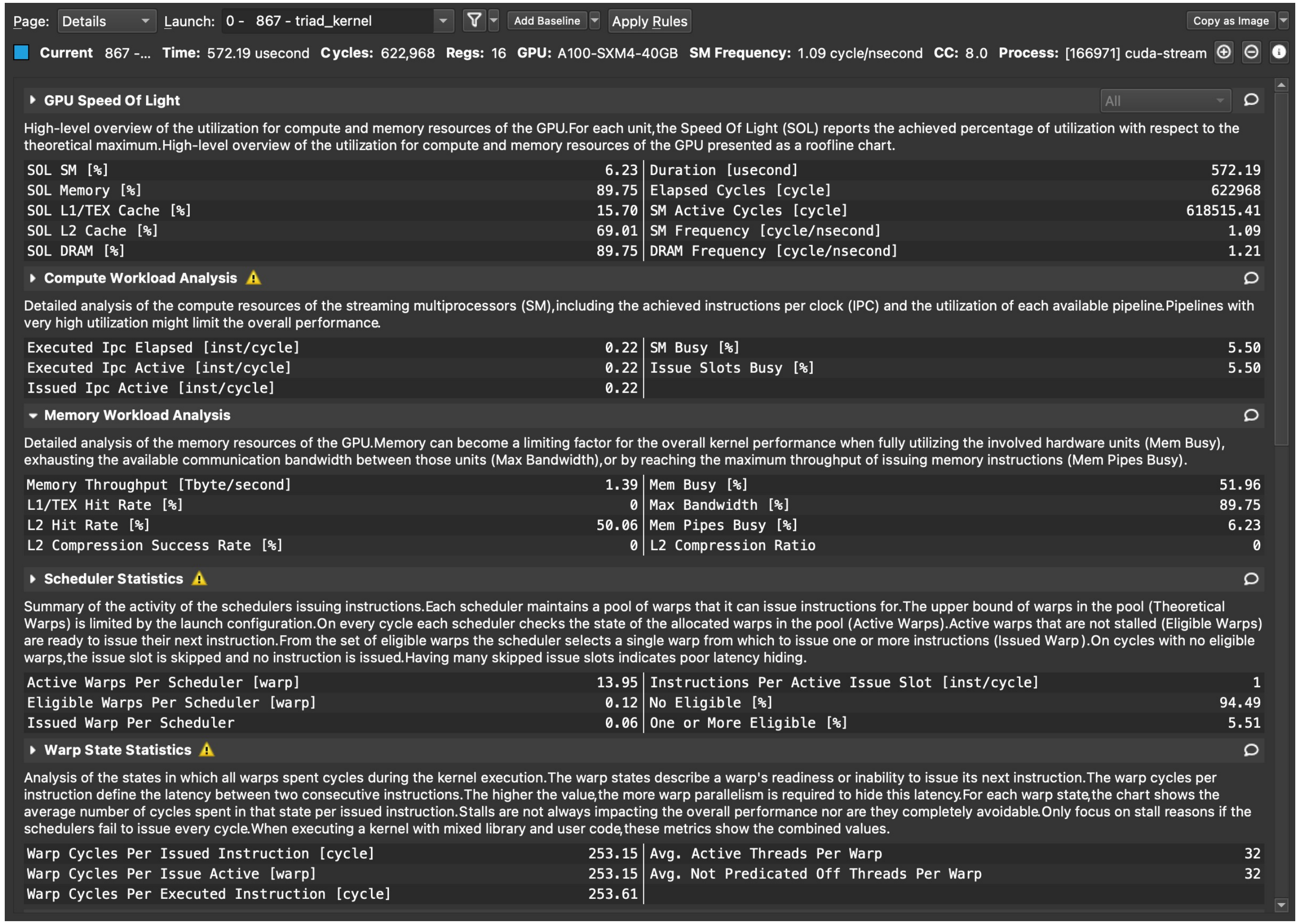


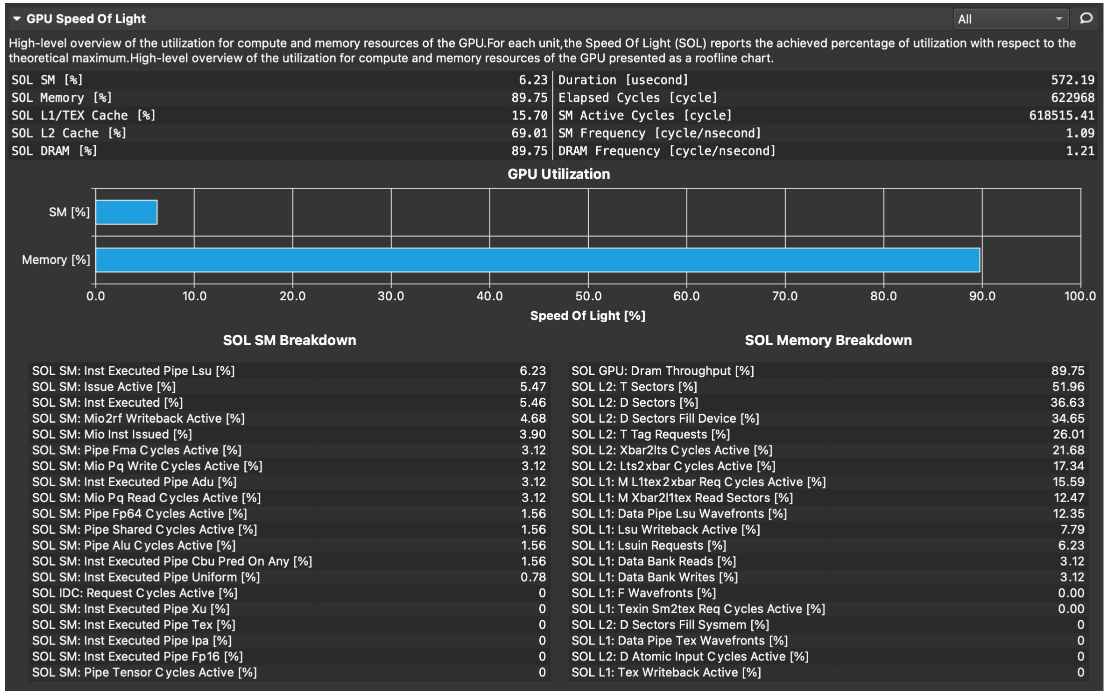
Nsight Compute with Stream benchmark

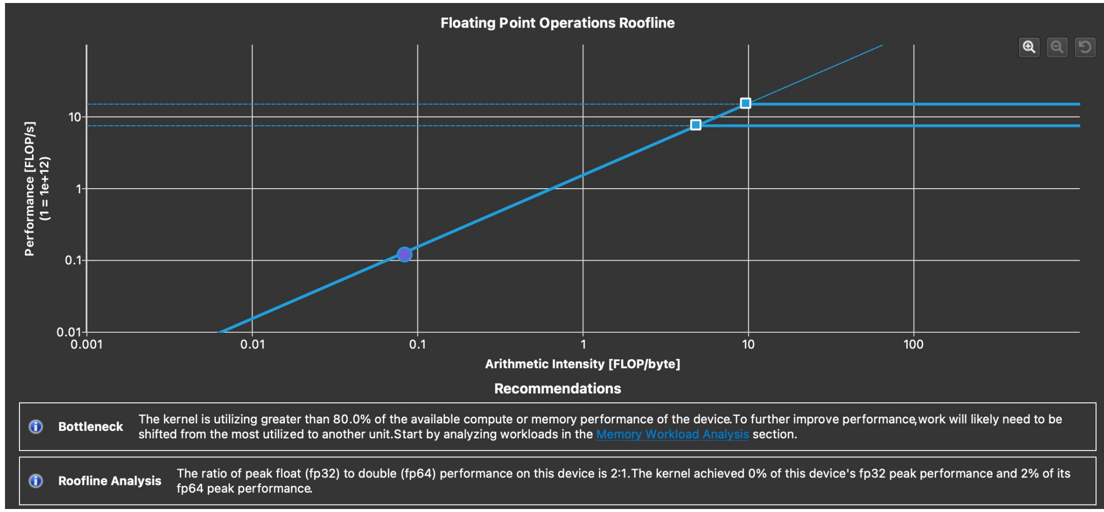
```
$ ncu --set detailed -k reg:"triad|add" -o JKreport-ncu_detailed-triad-add-BableStream ./cuda-stream
BabelStream
Version: 3.4
Implementation: CUDA
Running kernels 100 times
Precision: double
Array size: 268.4 MB (=0.3 GB)
Total size: 805.3 MB (=0.8 GB)
==PROF== Connected to process 166971 (/gpfs/mira-home/jkwack/HPC_benchmarks/BabelStream/JK_thetaGPU/cuda-stream)
Using CUDA device A100-SXM4-40GB
Driver: 11000
==PROF== Profiling "add_kernel": 0%....50%....100% - 18 passes
==PROF== Profiling "triad_kernel": 0%....50%....100% - 18 passes
==PROF== Profiling "add_kernel": 0%....50%....100% - 18 passes
==PROF== Profiling "triad_kernel": 0%....50%....100% - 18 passes
…
==PROF== Profiling "triad_kernel": 0%....50%....100% - 18 passes
Function    MBytes/sec  Min (sec)   Max         Average
Copy        1328115.853 0.00040     0.00042     0.00041
Mul         1302136.580 0.00041     0.00043     0.00042
Add         945.574     0.85166     1.15119     0.91029
Triad       943.893     0.85318     1.12513     0.95746
Dot         842990.134  0.00064     0.00074     0.00067
==PROF== Disconnected from process 2627262

…
```

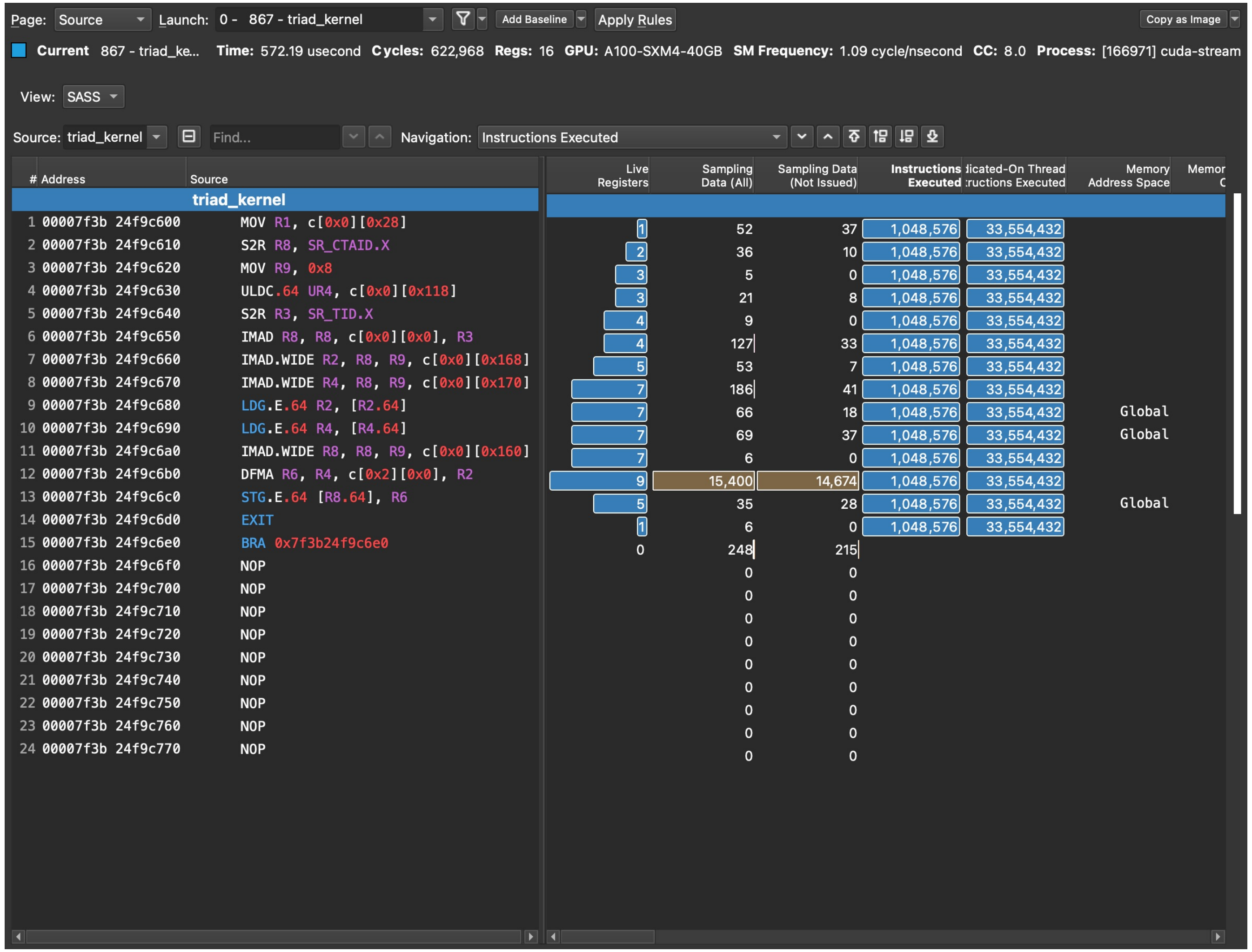
Nsight Compute w/ Stream

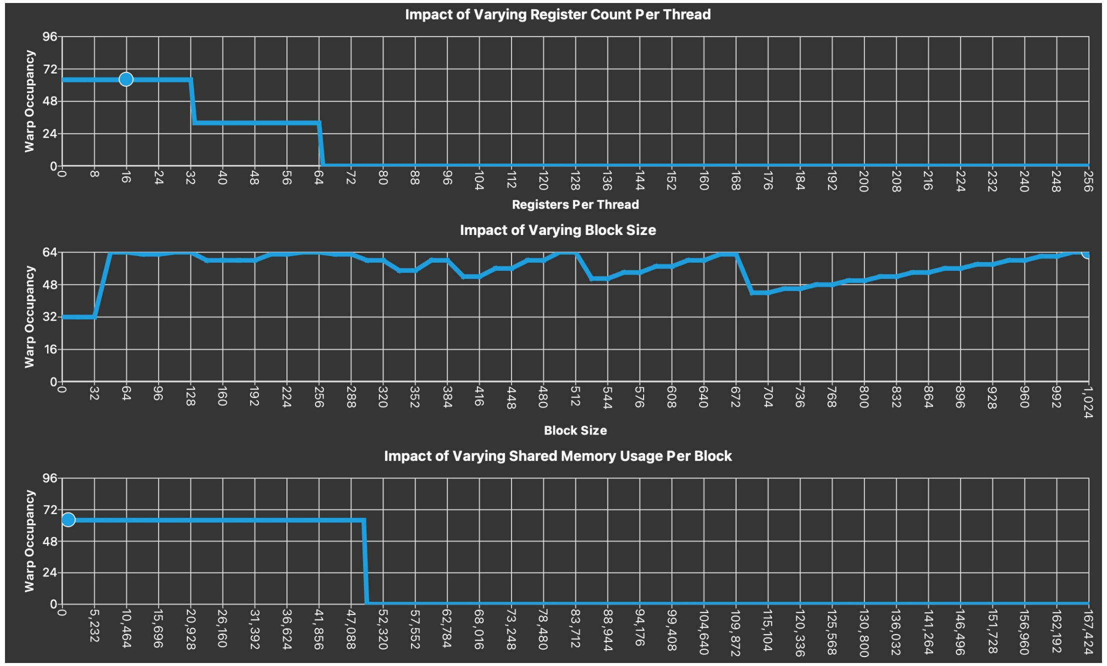


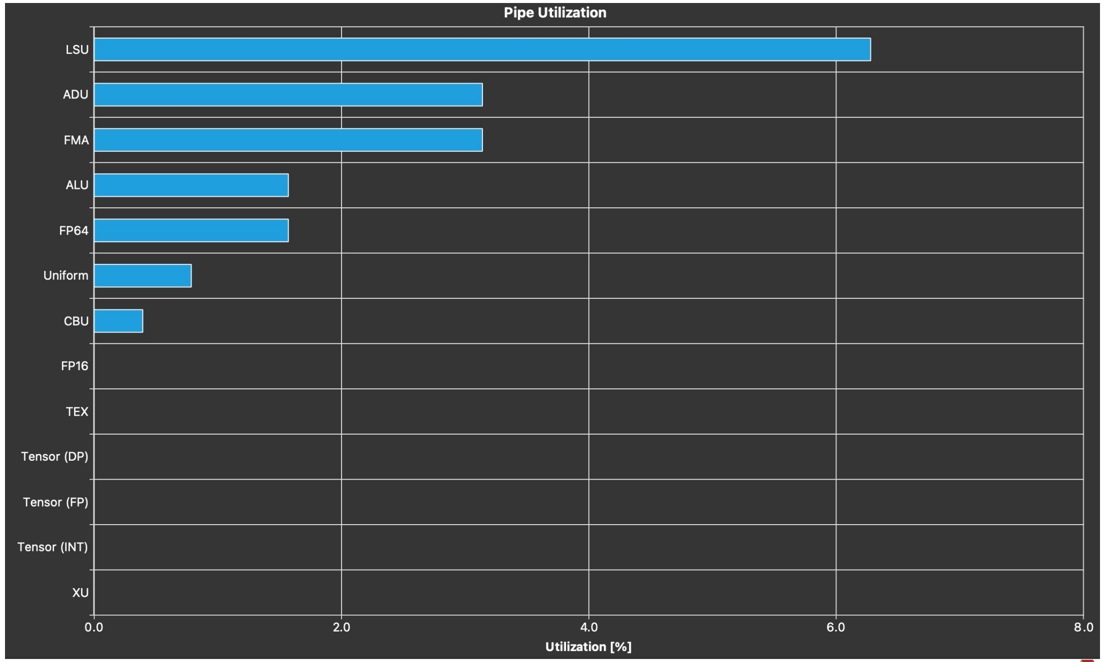




Nsight Compute w/ Stream







Argonne NATIONAL LABORATORY

Items for demo

- nsys results
  - Running in the terminal
  - Reviewing report on local client
  - Kernels
    - Shows in event view
    - Description in event view
  - CUDA Memory Operations
- ncu results
  - Running in the terminal
  - Reviewing report on local client
  - Baselines for comparison of multiple kernels
  - Arithmetic Intensity of add and triad kernels on roofline analysis
  - ADD/FMA instructions in SASS

Case 2: Geometric Series

GeoSeries Benchmark

- Computing geometric series (i.e., $S = 1+r+r^2+r^3+\dots+r^{nGeo}$)
  - Input parameters
    - n^2: array size
    - nGeo: the order of geometric series
  - Create a n^2 array (i.e., GeoR) with ratios: initialized by positive values less than 1.0
  - Repeat 100 times computation of geometric series (a n^2 array, GeoResult) with the given nGeo
- Build for A100
  - Using *nvhpc* module for OpenMP Target offloading
  - CFLAG for *nvc* compiler: *-mp=gpu -g -fast -O3*
- Comp_Geo kernel

```
#pragma omp target teams distribute parallel for collapse(2)
   for(j=0;j<n;j++){ for(i=0;i<n;i++){
         id = i+j*n;
         tmpR = GeoR[id];          tmpResult = 1.0E0;
         for (iGeo=1;iGeo<=nGeo;iGeo++){
            tmpResult = 1.0E0 + tmpR*tmpResult; }
         GeoResult[id] = tmpResult;
   }}
```

Offloading to GPU

Read one variable

nGeo times FMA instructions

Write one variable

Theoretical Arithmetic Intensity for DP = 2*nGeo/(2*8)

Performance of the benchmark on A100

Array size	Precison	nGEO	Wall time	FLOPs	Efficiency
8192^2	DP	10	0.0088 s	1.52 TF/s	15.7%
8192^2	DP	100	0.0201 s	6.67 TF/s	68.8%
8192^2	DP	1000	0.1769 s	7.59 TF/s	78.2%
8192^2	DP	10000	1.3921 s	9.64 TF/s	99.4%
8192^2	SP	10	0.0072 s	1.86 TF/s	9.5%
8192^2	SP	100	0.0128 s	10.46 TF/s	53.6%
8192^2	SP	1000	0.0944 s	14.22 TF/s	72.9%
8192^2	SP	10000	0.7444 s	18.03 TF/s	92.5%

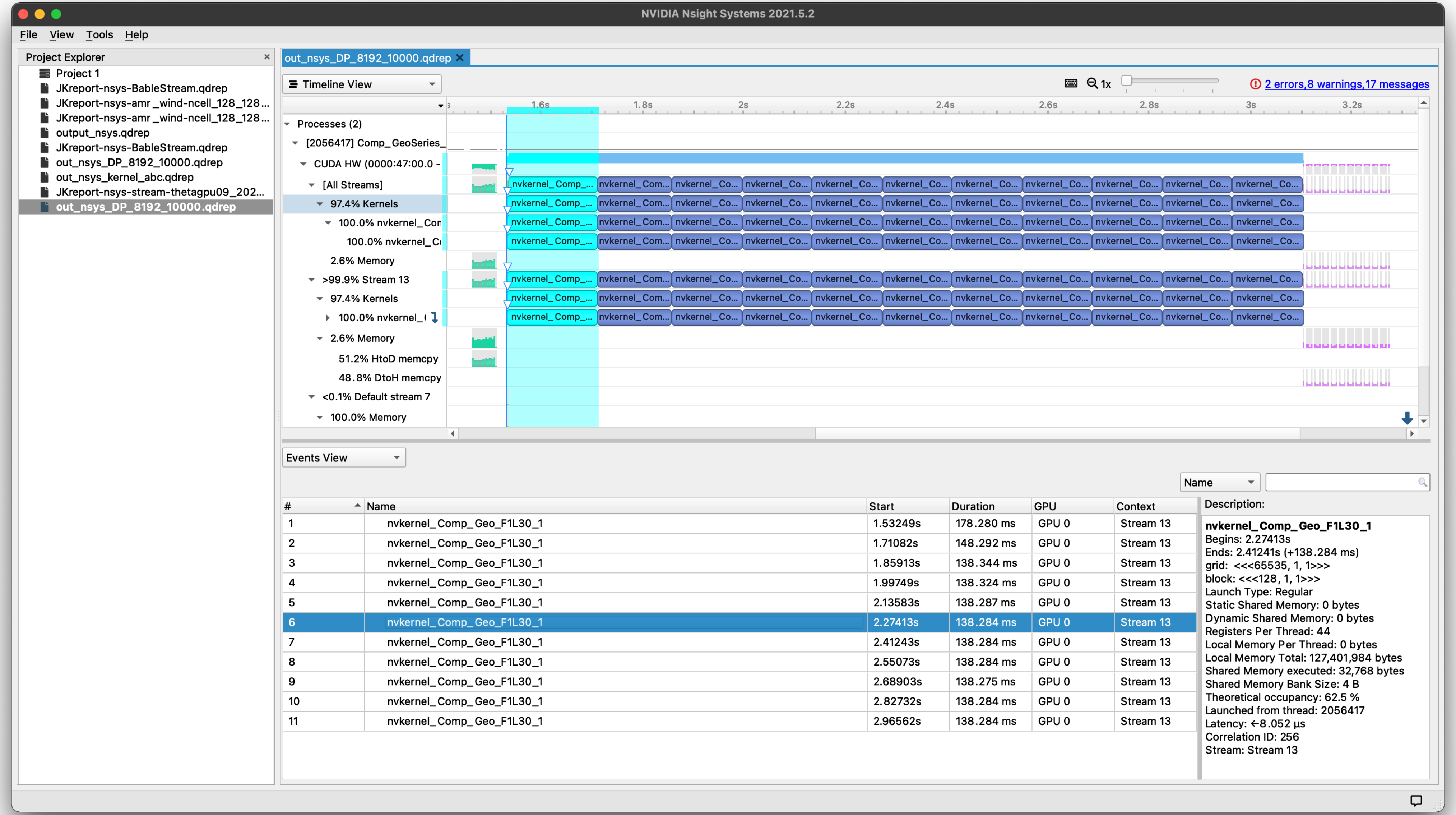
Nsight Systems with GeoSeries benchmark

```
$ nsys profile -o out_nsys_DP_8192_10000 --stats=true --force-overwrite true ./Comp_GeoSeries_omp_nvc_DP  8192 10000
…
CUDA API Statistics:
 Time(%)  Total Time (ns)  Num Calls     Average      Minimum      Maximum       StdDev             Name
 -------  ---------------  ---------  -------------  ----------  -----------  ------------  --------------------
    96.1    1,571,893,187         13  120,914,860.5       7,093  178,287,856  54,664,365.4  cuStreamSynchronize
     1.3       20,841,519         94      221,718.3       2,455      883,983     309,913.2  cuEventSynchronize
     1.3       20,608,269          1   20,608,269.0  20,608,269   20,608,269           0.0  cuMemAllocManaged
     1.1       17,893,965          1   17,893,965.0  17,893,965   17,893,965           0.0  cuMemHostAlloc
…
CUDA Kernel Statistics:
 Time(%)  Total Time (ns)  Instances     Average       Minimum      Maximum       StdDev                Name
 -------  ---------------  ---------  -------------  -----------  -----------  ------------  -------------------------
   100.0    1,571,221,933         11  142,838,357.5  138,274,948  178,280,096  12,131,315.9  nvkernel_Comp_Geo_F1L30_1

CUDA Memory Operation Statistics (by time):
 Time(%)  Total Time (ns)  Operations    Average    Minimum  Maximum   StdDev        Operation
 -------  ---------------  ----------  ---------  -------  -------  ---------  ------------------
    51.2       21,839,952          33  661,816.7    3,072  954,008  133,998.1  [CUDA memcpy HtoD]
    48.8       20,823,032          33  631,001.0    4,768  867,288  118,995.5  [CUDA memcpy DtoH]

CUDA Memory Operation Statistics (by size in KiB):
    Total     Operations    Average    Minimum   Maximum     StdDev        Operation
 -----------  ----------  ----------  -------  ----------  ---------  ------------------
 524,288.000          33  15,887.515    0.008  16,384.000  2,852.087  [CUDA memcpy HtoD]
 524,287.992          33  15,887.515    1.992  16,383.938  2,851.731  [CUDA memcpy DtoH]
…
```

Nsight Systems with GeoSeries benchmark

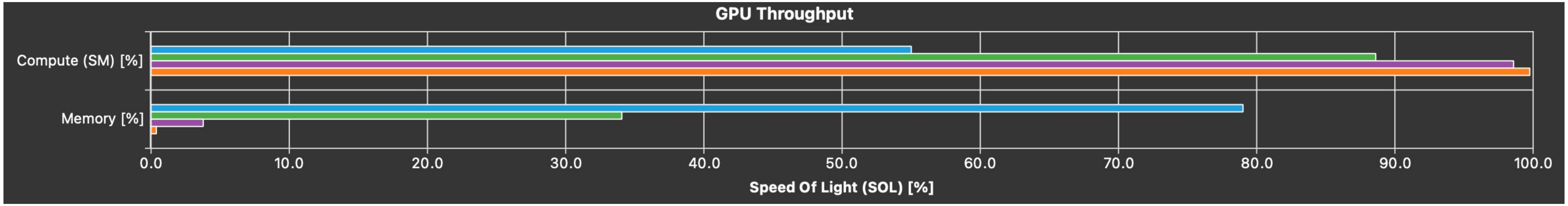


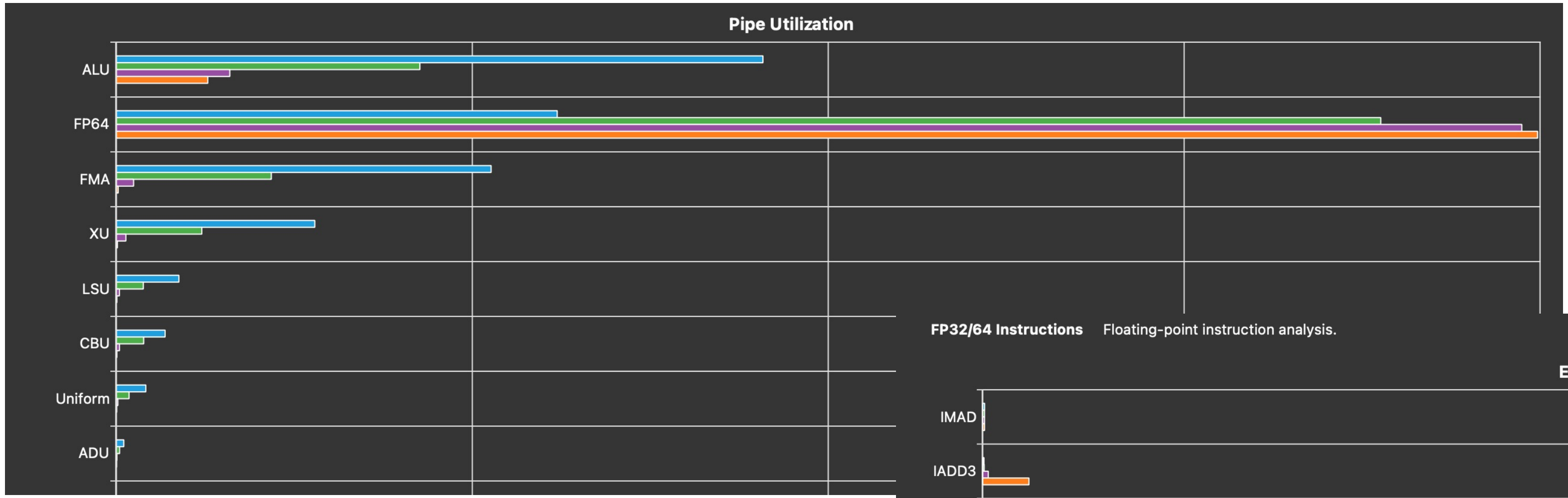
Nsight Compute with GeoSeries benchmark

```
$ ncu --set detailed -k regex:"Comp_" -o out_ncu_DP_8192_10 ./Comp_GeoSeries_omp_nvc_DP 8192 10
…
                                Number of MPI process:      1
                                            Precision: double
                 Number of rows/columns of the matrix:   8192
                 The highest order of geometric series:     10
                               Number of repetitions:     10
                           Memory Usage per MPI rank:  1073.741824 MB
                              Warming up .....
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
                              Main Computations  10 repetitions ......
                              0%                    25%                       50%                      75%                      100%
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
==PROF== Profiling "nvkernel_Comp_Geo_F1L30_1": 0%....50%....100% - 18 passes
                              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
                                            Error_MPI_{Min,Mean,Max}/MPI =   1.4900e-08    1.4900e-08    1.4900e-08
                                           GFLOP-rate_{Min,Mean,Max}/MPI =     1.251655      1.251655      1.251655
                                                               Wall Time =    10.723224 sec
                                                               FLOP-rate =     1.251655 GFLOP/sec
==PROF== Disconnected from process 2364879
…
```

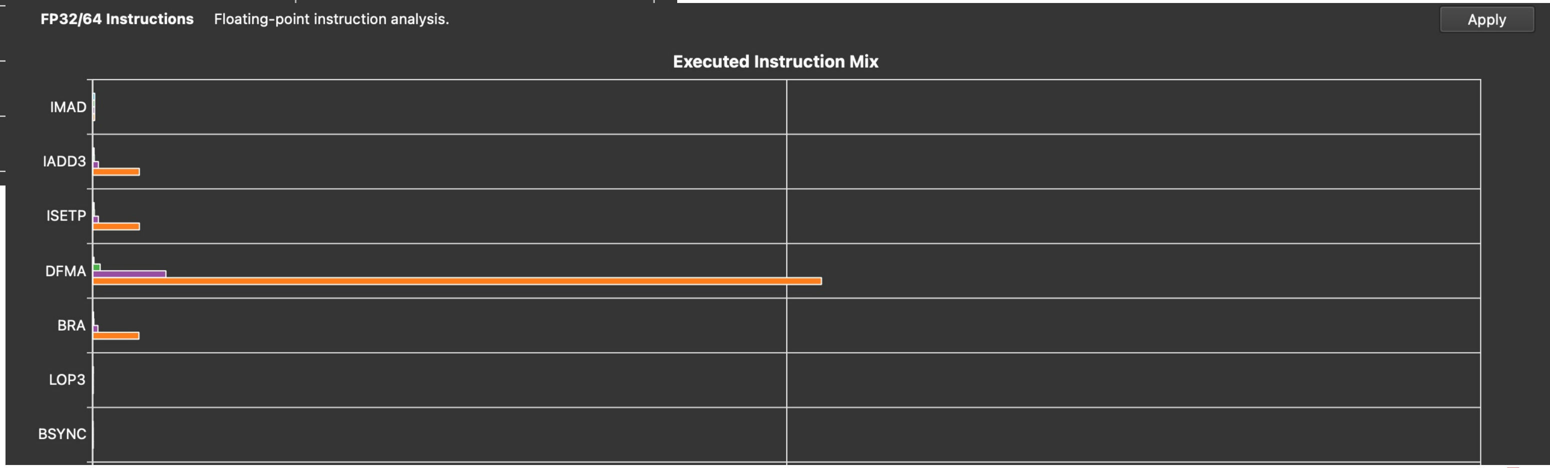
Nsight Compute w/ GeoSeries

nGEO=10 (blue)
nGEO=100 (green)
nGEO=1000 (purple)
nGEO=10000 (orange)





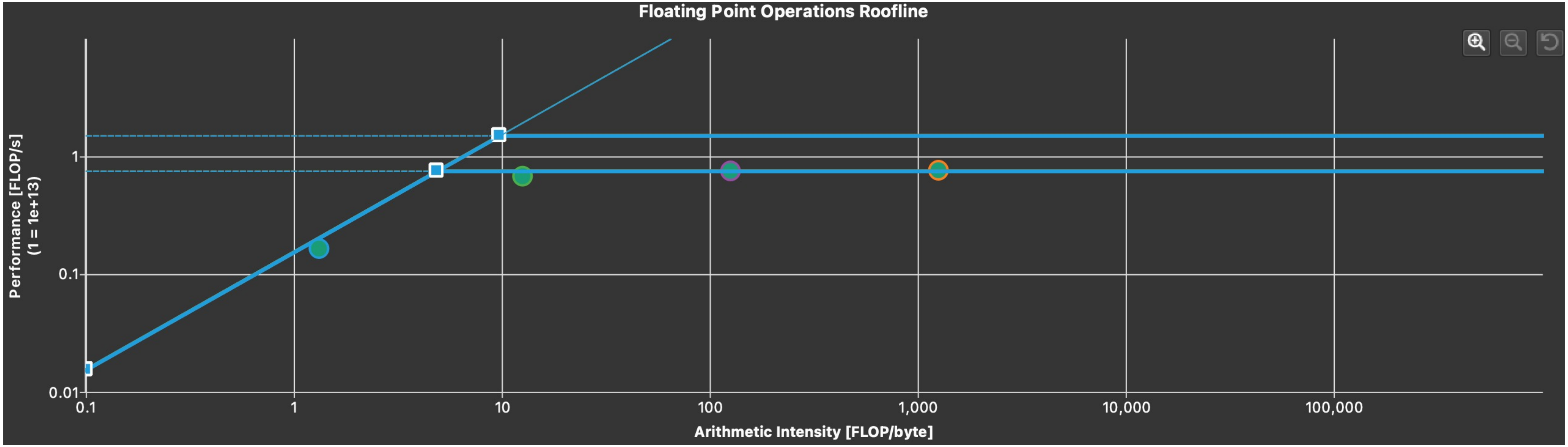
ALU: Arithmetic Logic Unit
FP64: Double-precision floating point unit
FMA: FP32 FMA pipeline



DFMA: FP64 Fused Multiply Add

Nsight Compute w/ GeoSeries

nGEO=10 (blue)
nGEO=100 (green)
nGEO=1000 (purple)
nGEO=10000 (orange)



Precision	nGEO	Theoretical A/I	Measured A/I	Measured FLOPs
DP	10	1.25	1.33	1.63 TF/s
DP	100	12.5	12.68	6.71 TF/s
DP	1000	125	126.75	7.44 TF/s
DP	10000	1250	1266.92	7.53 TF/s

Items for demo

- nsys results
  - Reviewing report on local client
  - Kernels
    - Shows in event view
    - Description in event view
  - CUDA Memory Operations
- ncu results
  - Reviewing report on local client
  - Baselines for comparison of multiple executions
  - Details: GPU Throughput
  - Details: Pipe Utilization
  - Details: Executed Instruction Mix
  - Details: Floating Point Operation Roofline
  - Source: SASS
    - nGEO=10: DFMA (stall long scoreboard)
    - nGEO=10000: DFMA
  - FMA instructions in SASS

Thank you!