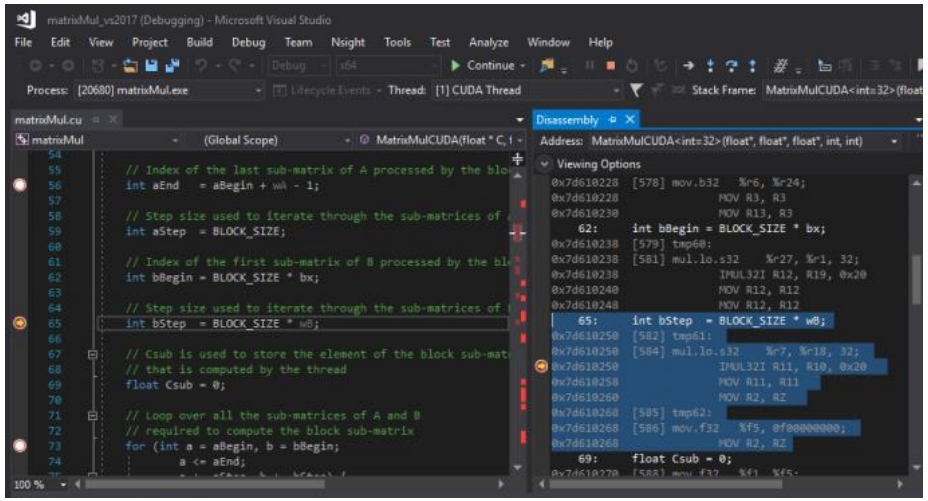
NVIDIA Developer Tools

Kristopher Keipert
Solutions Architect, NVIDIA

ECP Argonne NATIONAL LABORATORY

DEVELOPER TOOLS

Debuggers: cuda-gdb, Nsight Visual Studio Edition

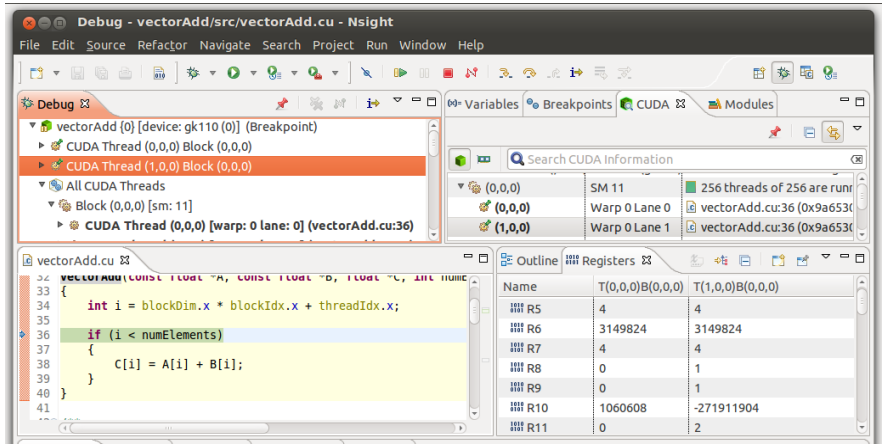


Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)

Correctness Checker:: Compute Sanitizer

```
$ compute-sanitizer --leak-check full memcheck_demo
========= COMPUTE-SANITIZER
Mallocing memory
Running unaligned_kernel
Ran unaligned_kernel: no error
Sync: no error
Running out_of_bounds_kernel
Ran out_of_bounds_kernel: no error
Sync: no error
========= Invalid __global__ write of size 4 bytes
=========     at 0x60 in memcheck_demo.cu:6:unaligned_kernel(void)
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x400100001 is misaligned
```

IDE integrations: Nsight Eclipse Edition
Nsight Visual Studio Edition
Nsight Visual Studio **Code** Edition



NVIDIA

PROGRAMMING THE NVIDIA PLATFORM

CPU, GPU, and Network

ACCELERATED STANDARD LANGUAGES

ISO C++, ISO Fortran

```
std::transform(par, x, x+n, y, y,
    [=](float x, float y){ return y +
a*x; }
);
```

```
do concurrent (i = 1:n)
   y(i) = y(i) + a*x(i)
enddo
```

```
import cunumeric as np
…
def saxpy(a, x, y):
    y[:] += a*x
```

INCREMENTAL PORTABLE OPTIMIZATION

OpenACC, OpenMP

```
#pragma acc data copy(x,y) {
...
std::transform(par, x, x+n, y, y,
    [=](float x, float y){
        return y + a*x;
});
...
}

#pragma omp target data map(x,y) {
...
std::transform(par, x, x+n, y, y,
    [=](float x, float y){
        return y + a*x;
});
...
}
```

PLATFORM SPECIALIZATION

CUDA

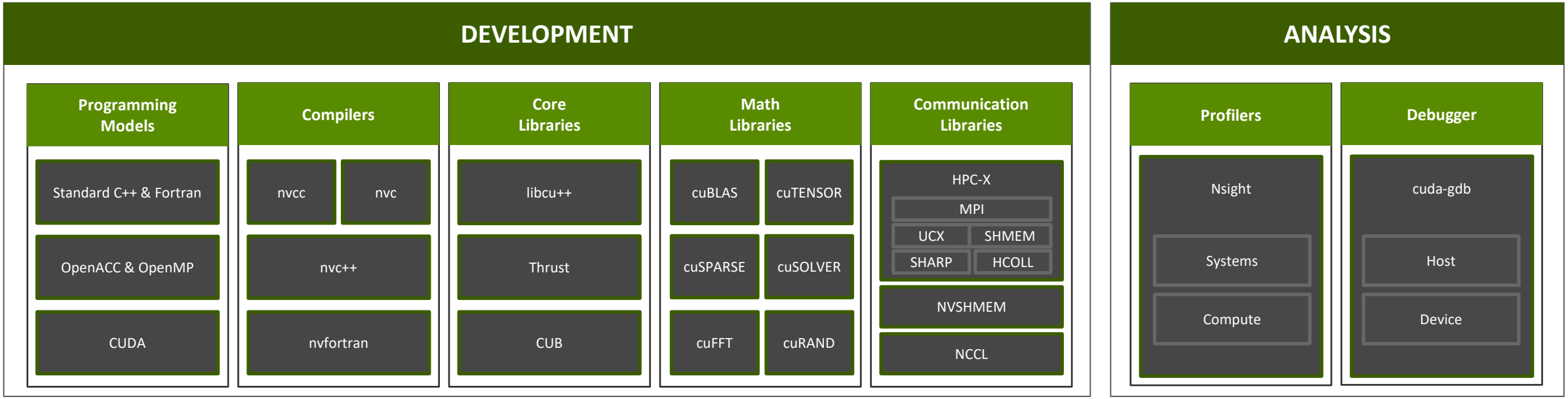
```
__global__
void saxpy(int n, float a,
           float *x, float *y) {
  int i = blockIdx.x*blockDim.x +
          threadIdx.x;
  if (i < n) y[i] += a*x[i];
}

int main(void) {
  ...
  cudaMemcpy(d_x, x, ...);
  cudaMemcpy(d_y, y, ...);

  saxpy<<<(N+255)/256,256>>>(...);

  cudaMemcpy(y, d_y, ...);
```

ACCELERATION LIBRARIES

Core	Math	Communication	Data Analytics	AI	Quantum

NVIDIA HPC SDK

Available at developer.nvidia.com/hpc-sdk, on NGC, via Spack, and in the Cloud

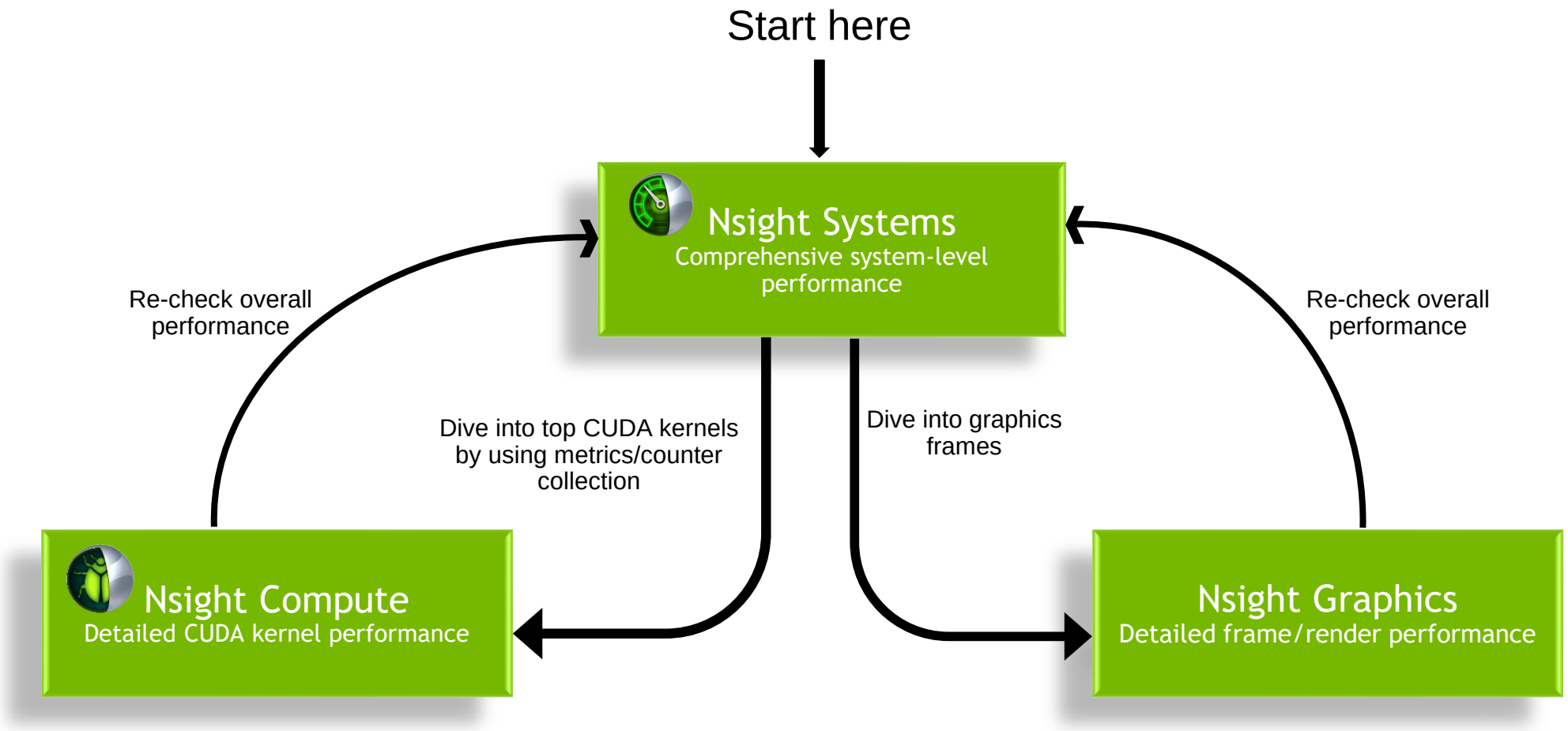


Develop for the NVIDIA Platform: GPU, CPU and Interconnect

Libraries | Accelerated C++ and Fortran | Directives | CUDA

7-8 Releases Per Year | Freely Available

NSIGHT TOOLS WORKFLOW



NSIGHT SYSTEMS

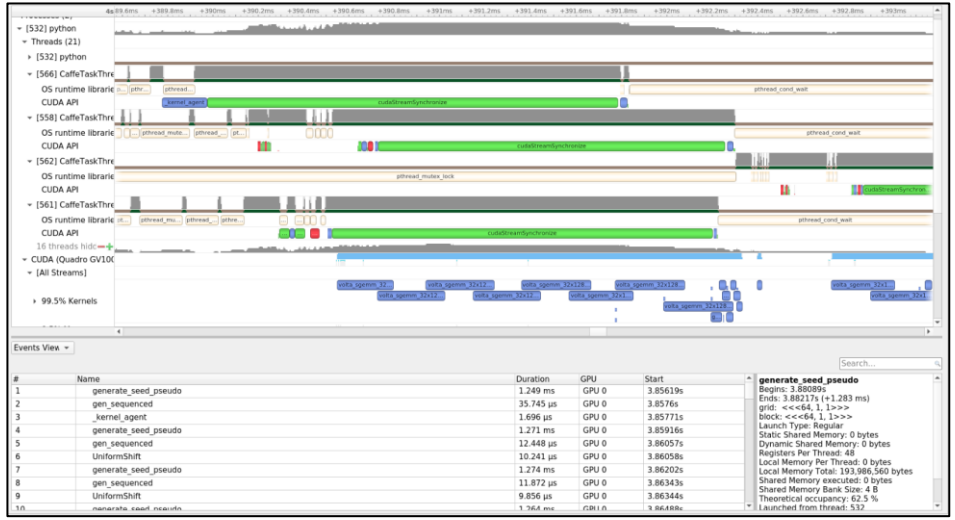
SYSTEM PROFILER

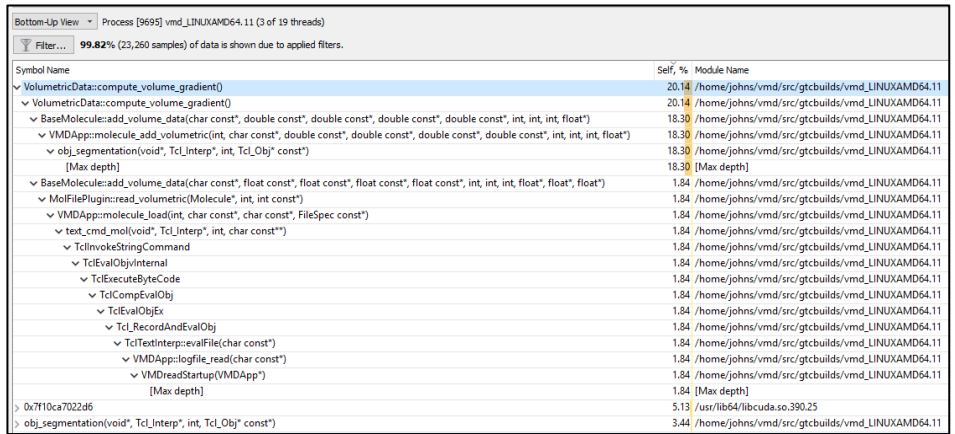
Key Features:

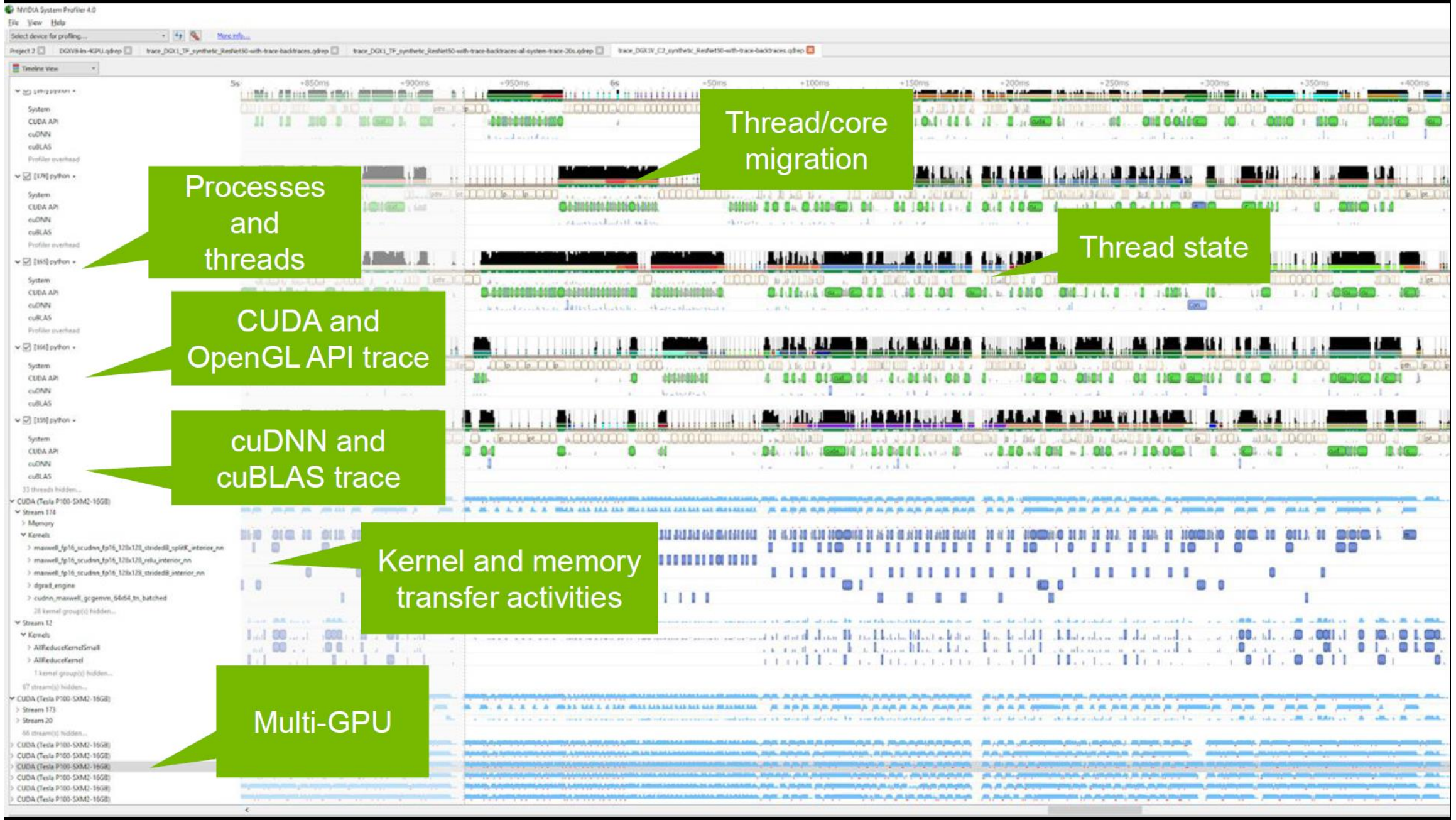
- System-wide application algorithm tuning
  - Multi-process tree support
- Locate optimization opportunities
  - Visualize millions of events on a very fast GUI timeline
  - Or gaps of unused CPU and GPU time
- Balance your workload across multiple CPUs and GPUs
  - CPU algorithms, utilization and thread state
    GPU streams, kernels, memory transfers, etc
- Command Line, Standalone, IDE Integration

OS: Linux (x86, Power, Arm SBSA, Tegra), Windows, MacOSX (host)
GPUs: Pascal+

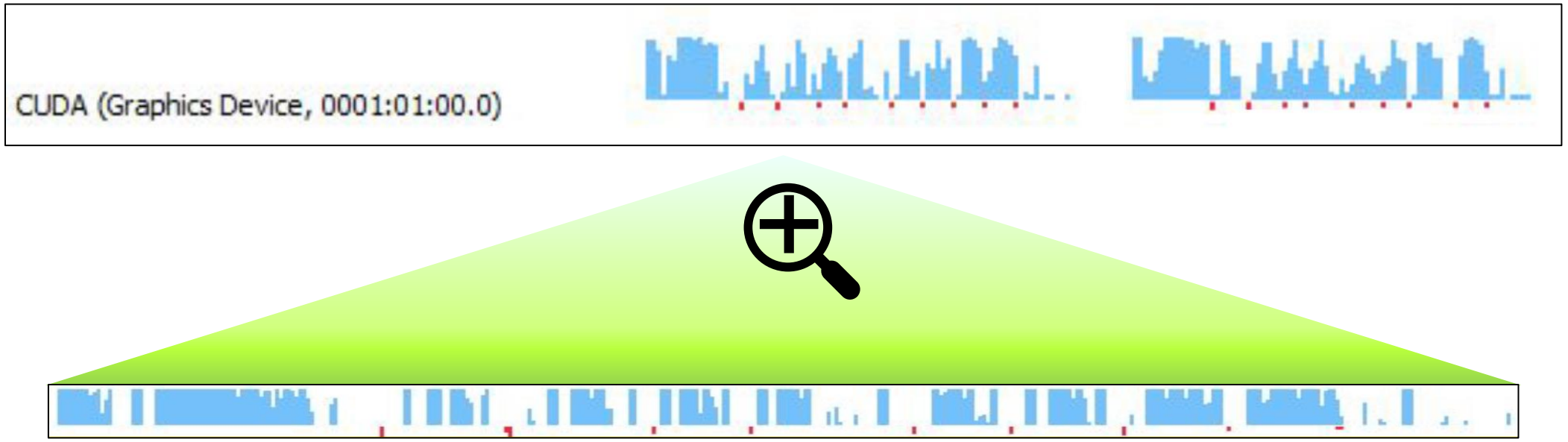
Docs/product: https://developer.nvidia.com/nsight-systems






Thread/core migration
Processes and threads
Thread state
CUDA and OpenGL API trace
cuDNN and cuBLAS trace
Kernel and memory transfer activities
Multi-GPU

READING UTILIZATION

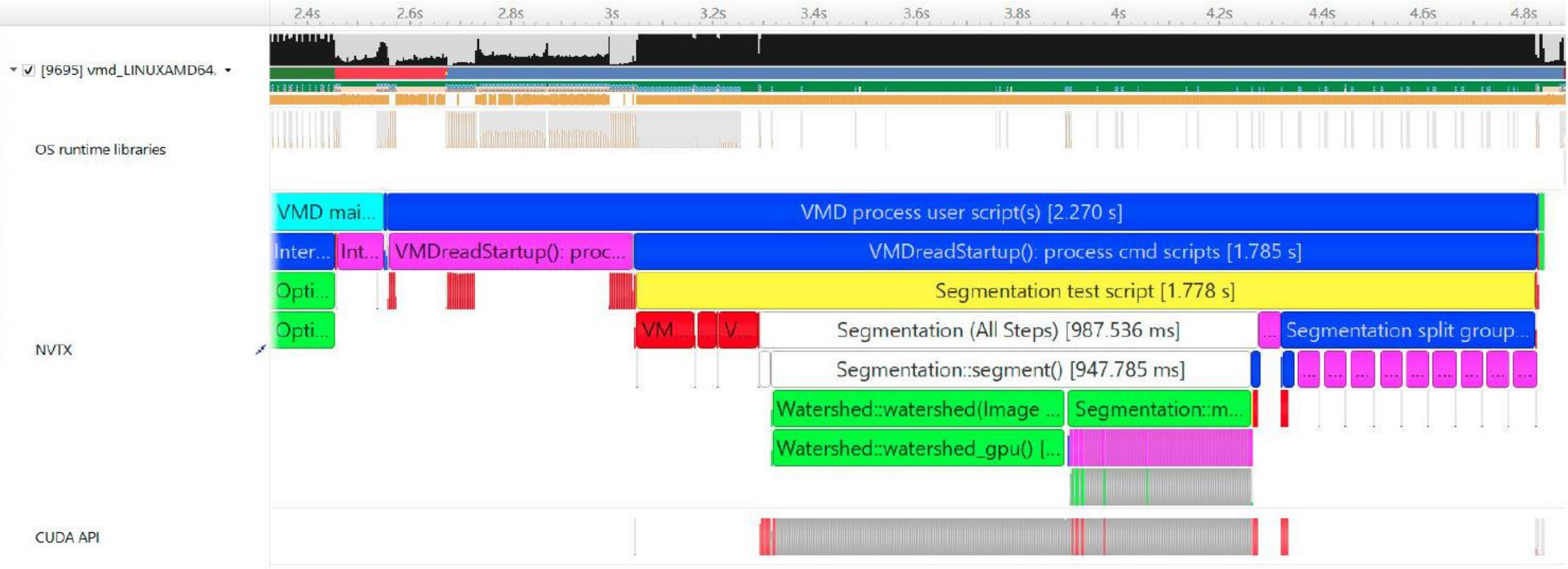


- Zoom in to valleys to find gaps!

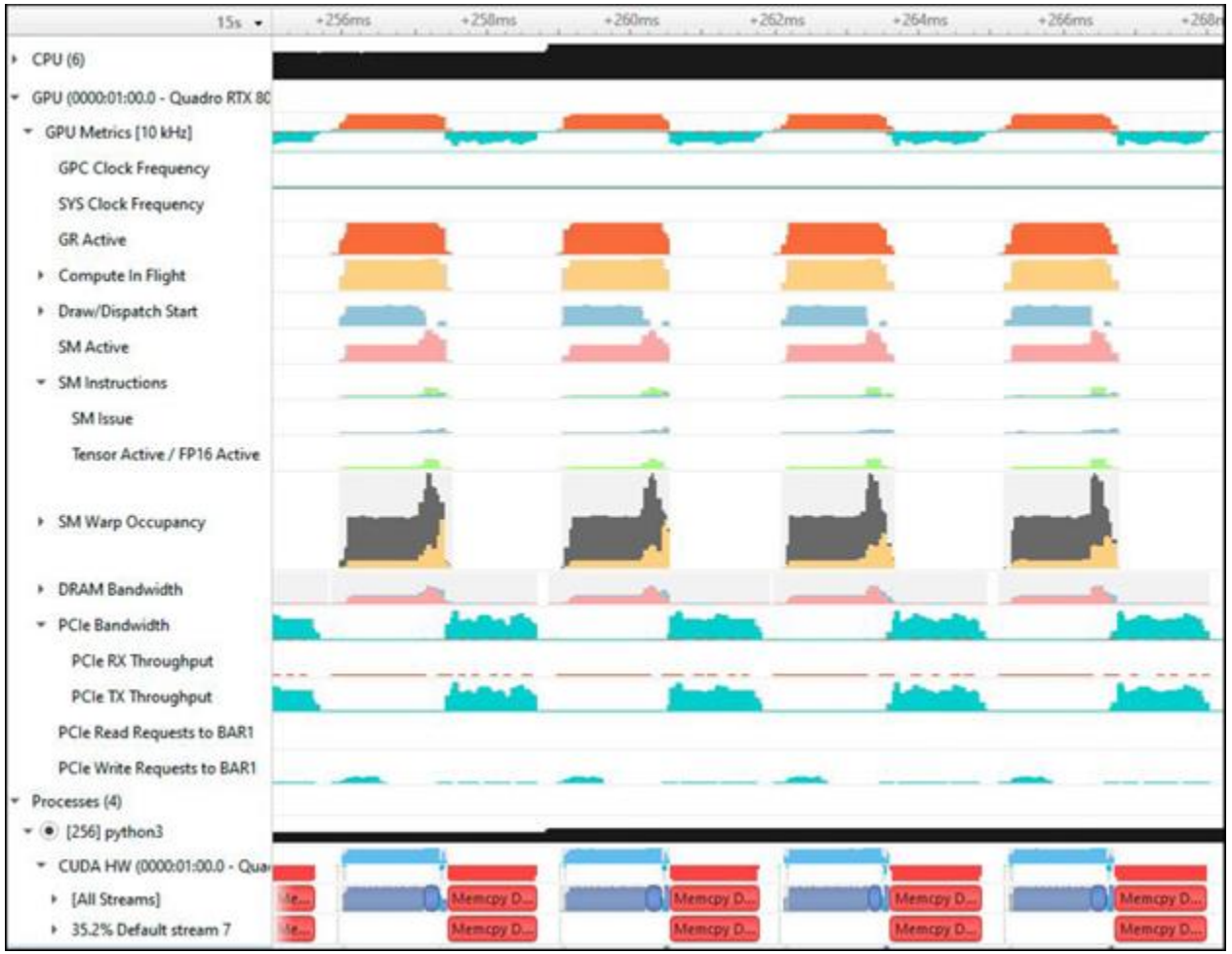
ZOOM/FILTER TO EXACT AREAS OF INTEREST

NVTX: NVIDIA TOOLS EXTENSIONS

Code Annotation API



GPU METRICS SAMPLING



- Useful GPU utilization metrics, but no kernel names / correlation

INTERPRETING GPU SAMPLING METRICS

- GR Activity -> GPU is doing work
  - SM, NVENC, NVDEC, Graphics
- SM Activity -> Utilizing width of GPU
  - If low, modify kernel grid dimension or increase batc size
- SM Instruction Issued -> GPU is performing lots of instructions
  - Stalled waiting on memory?
  - Not enough warps to cover memory latency? Issue larger kernel block dimensions.
- SM Instructions tensor activity -> Tensor core utilization
  - Performance up, SM instructions can drop (depending on arch)
  - Can be limited by shared memory, waiting for loads
- Note: Requires disabling ***DCGM*** and DL built-in profilers

APPLICATION PROFILES WITH NSIGHT SYSTEMS

```
$ nsys profile –o report –stats=true ./myapp.exe
```

- Generated file: report.qdrep (or report.nsys-rep)
  Open for viewing in the Nsight Systems UI

- When using MPI, recommended to use *nsys* after mpirun/srun:

```
$ mpirun –n 4 nsys profile ./myapp.exe
```

PROFILING DL MODELS

- Pytorch
  - DNN Layer annotations are disabled by default
  - ++ ”*with torch.autograd.profiler.emit_nvtx():*”
  - Manually with *torch.cuda.nvtx.range_(push/pop)*
  - TensorRT backend is already annotated

- Tensorflow
  - Annotated by default with NVTX in NVIDIA TF containers
  - TF_DISABLE_NVTX_RANGES=1 to disable for production

QUESTIONS TO GUIDE PROFILE ANALYSIS

- What is hot?
  - Can I make it faster, shrink the problem, parallelize it? (Not always…)
  - Reduce precision?

- What is cold?
  - Fill the gaps in the timeline
  - Can I take advantage of unused hardware?
  - Unnecessary dependencies or syncs?

GENERAL OPTIMIZATION TIPS

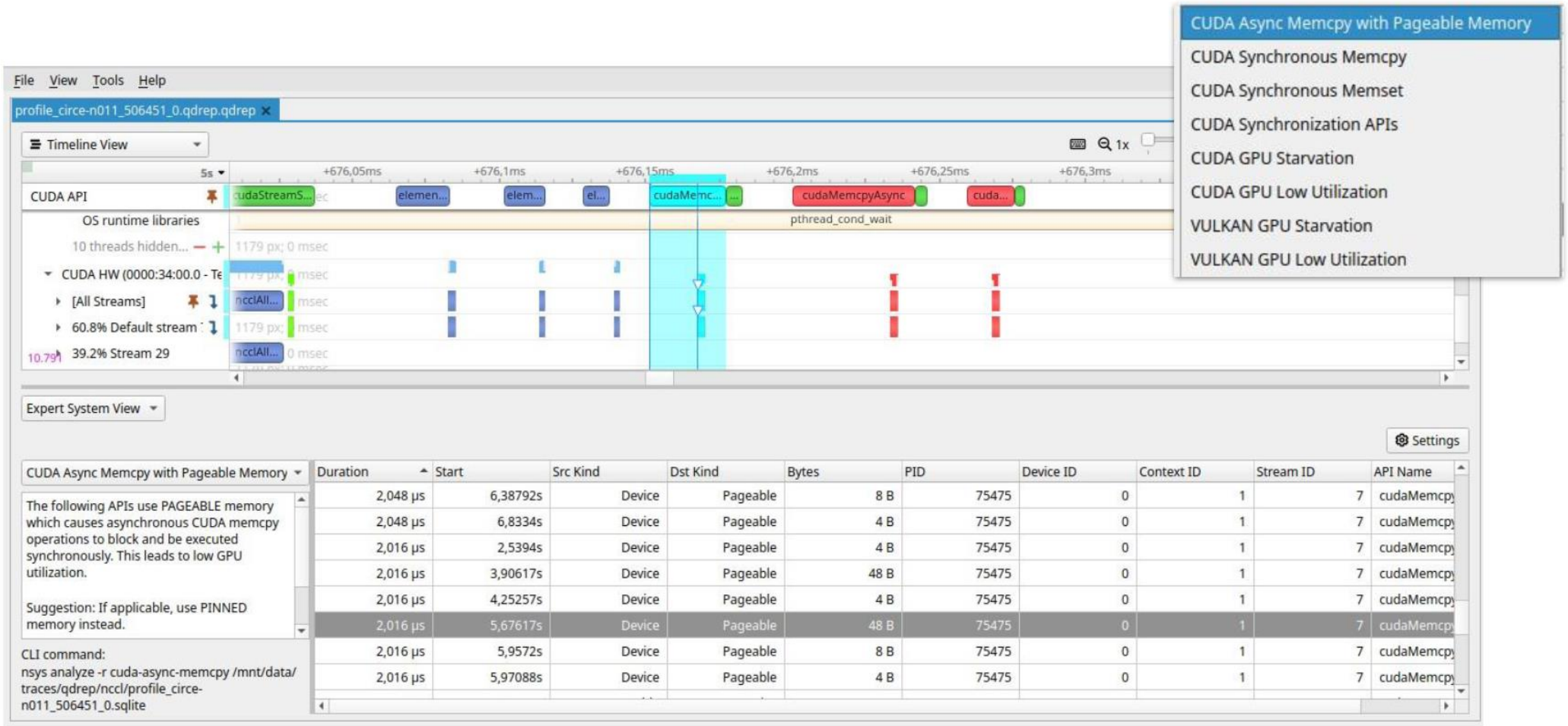
- Using tensor cores?
  - Minimize conversions/transposes
- Increase grid and batch size to utilize GPUs width
- Conventional parallelism – more worker threads!
- Parallel pipelining
  - No data dependency? Parallelize!
  - Prefetch next batch/iteration during computation
- Can I reorder sooner?

GENERAL OPTIMIZATION TIPS

- Fuse tiny kernels, copies, memsets.
  - Check out CUDA Graphs
- Overlap/oversubscribe with MPS
- Multi-buffering
  - Don’t make everyone wait on the same piece of memory
  - Double, triple buffer
- Avoid moving data back to the CPU
  - Pre-allocate and recycle!
- Minimize managed memory page faults
  - Prefetch!

EXPERT SYSTEMS & STATISTICS

BUILT-IN DATA ANALYTICS WITH ADVICE



NSIGHT COMPUTE

KERNEL PROFILING TOOL

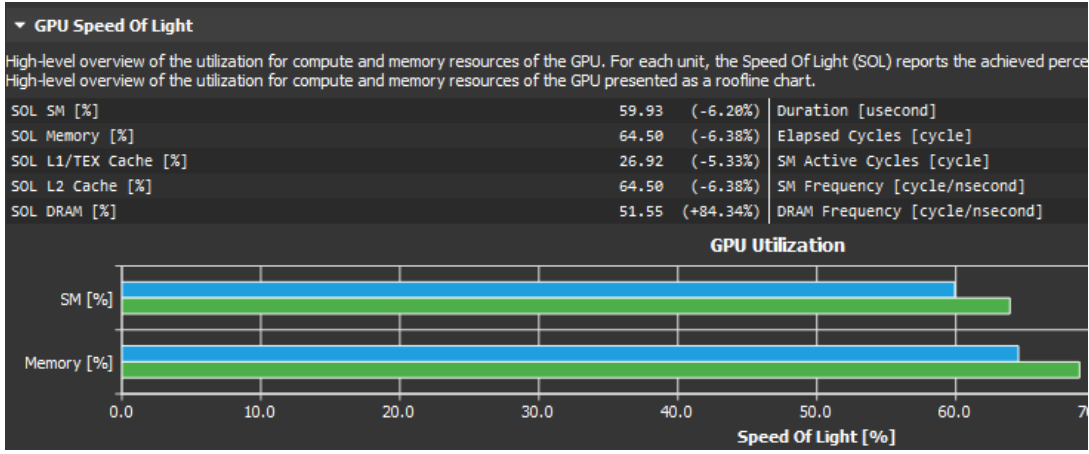
Key Features:

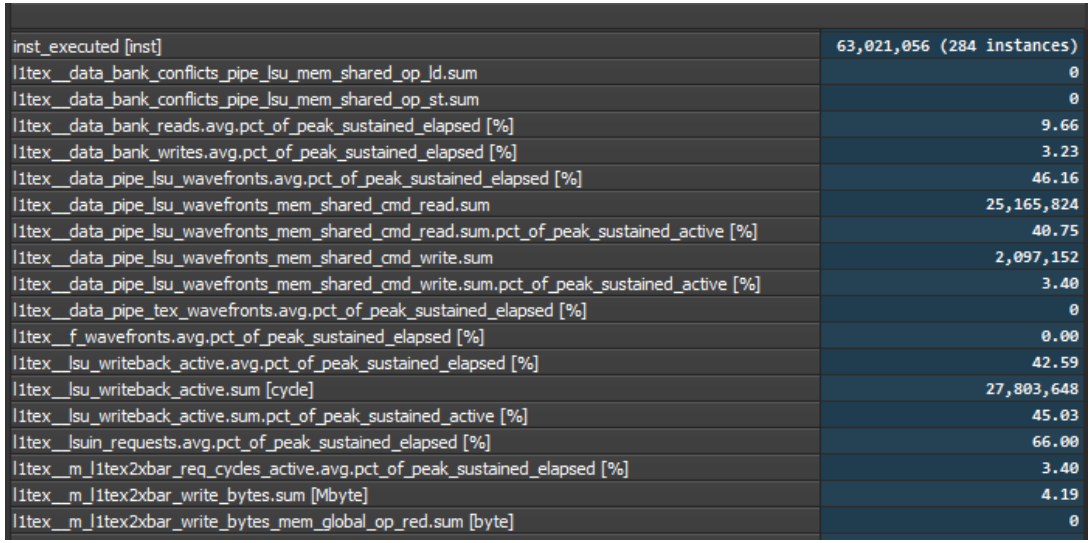
- Interactive CUDA API debugging and kernel profiling
- Built-in rules expertise
- Fully customizable data collection and display
- Command Line, Standalone, IDE Integration, Remote Targets

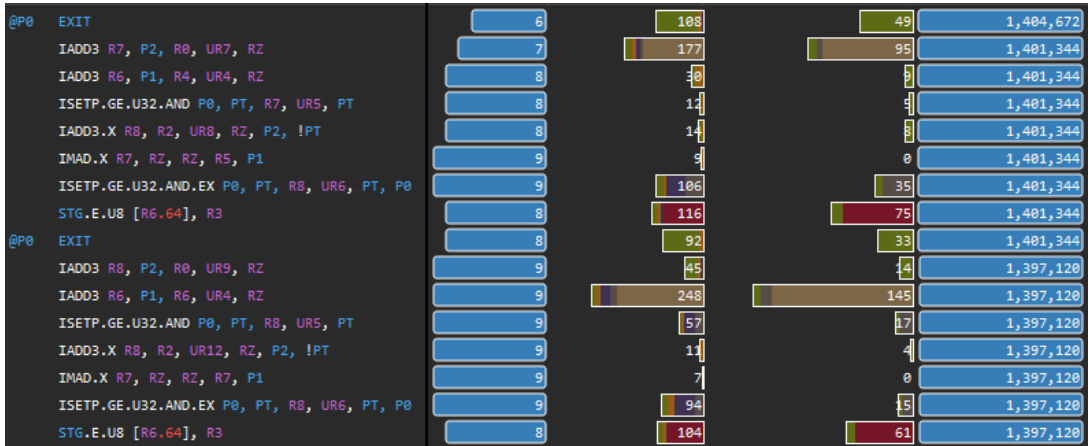
OS: Linux (x86, Power, Tegra, Arm SBSA), Windows, MacOSX (host only)

GPUs: Volta, Turing, Ampere GPUs

Docs/product: https://developer.nvidia.com/nsight-compute



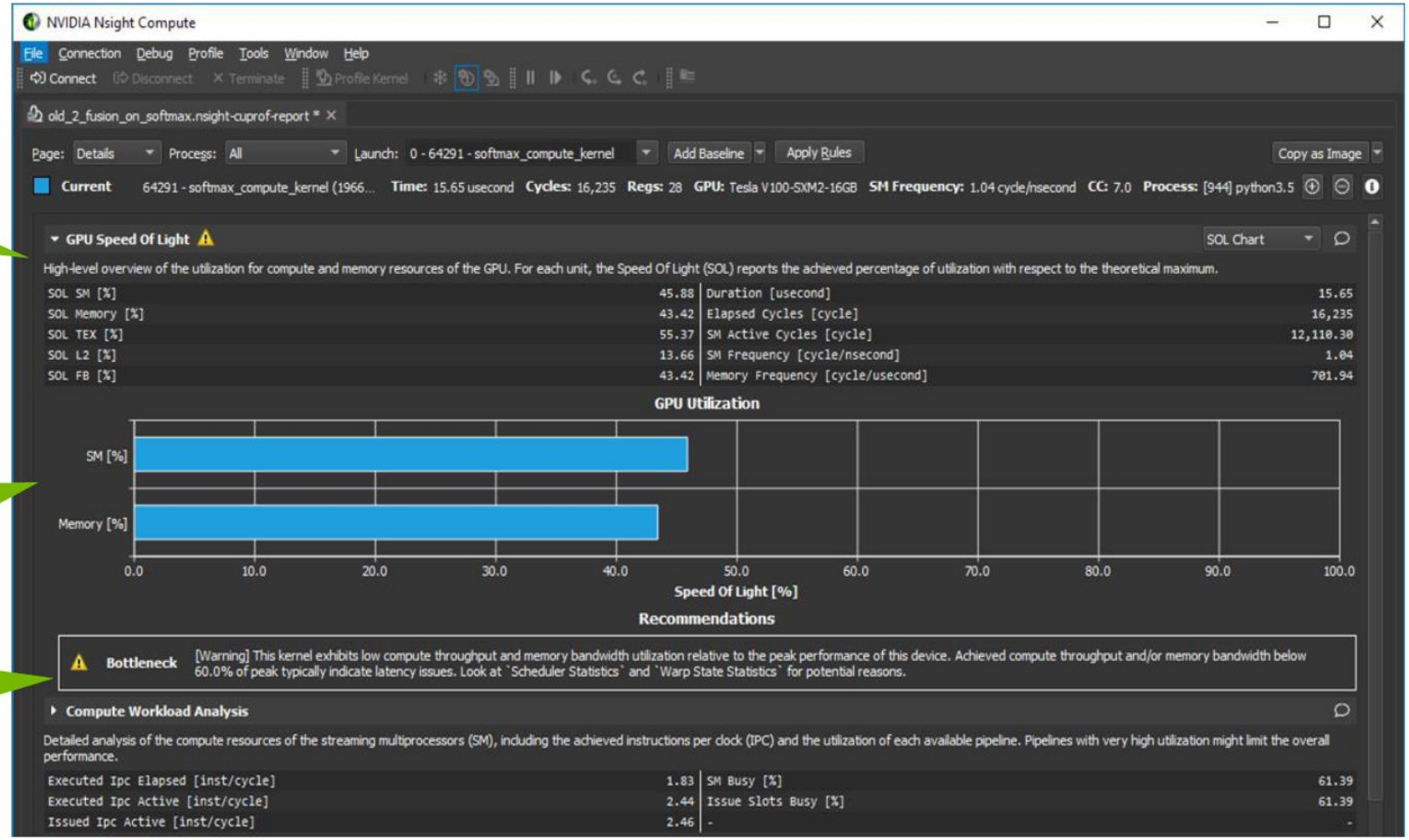




Targeted metric sections

Customizable data collection and presentation

Built-in expertise for Guided Analysis and optimization



Visual memory analysis chart

Metrics for peak performance ratios

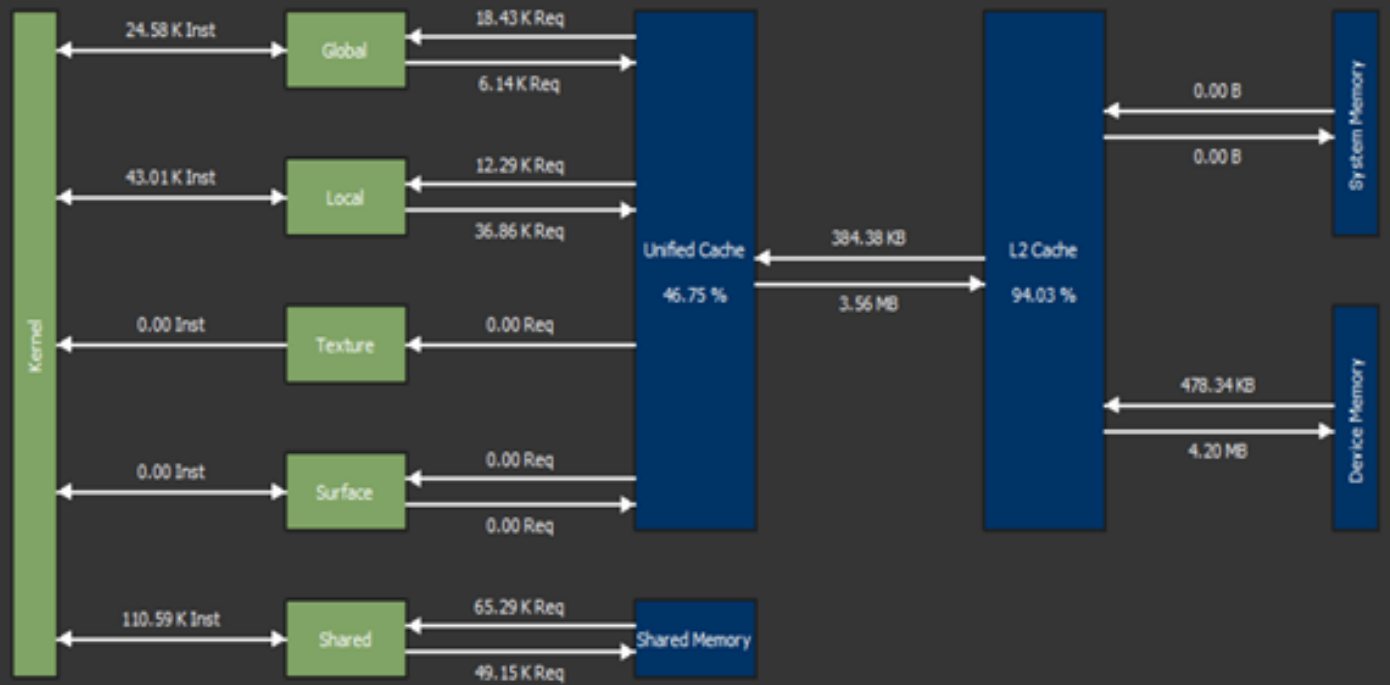
Memory Workload Analysis

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]	310.08	Mem Busy [%]	42.60
L1 Hit Rate [%]	46.75	Max Bandwidth [%]	44.73
L2 Hit Rate [%]	94.03	Mem Pipes Busy [%]	42.23

Memory Chart

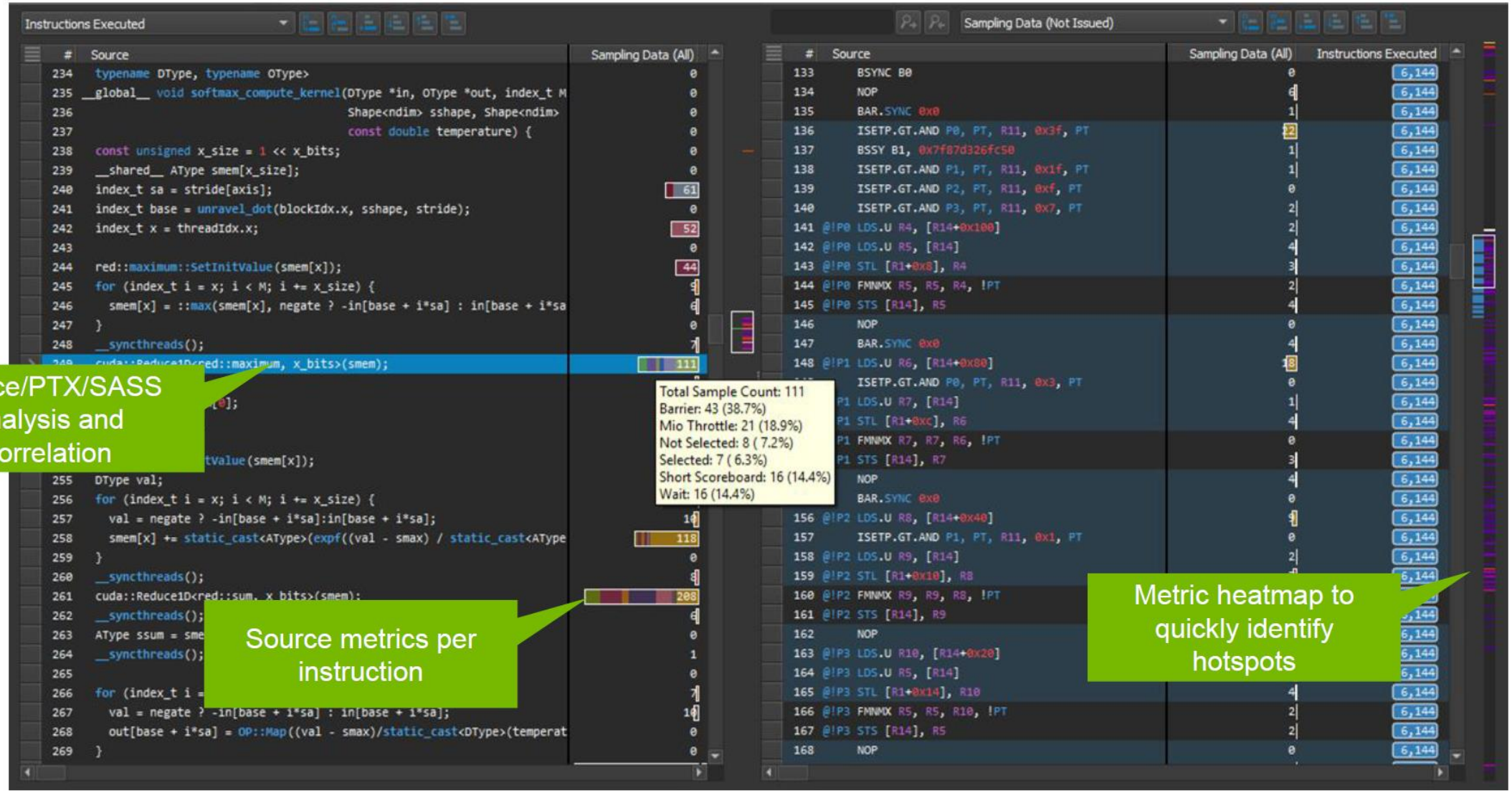


Shared Memory

	Instructions	Requests	% Peak	Bank Conflicts
Shared Load	61,440	65,289	6.59	3,698
Shared Store	49,152	49,152	4.96	0
Shared Atomic	0	-	-	-
Total	110,592	114,441	11.55	3,698

First-Level (Unified) Cache

	Instructions	SM->TEX Requests	% Peak	Hit Rate	TEX->L2 Requests	% Peak	L2->TEX Returns	% Peak
Global Load Cached	18,432	18,432	1.86	66.65	-	-		
Global Load Uncached	-	-	-	-	-	-		
Local Load Cached	12,288	12,288	1.24	100	-	-	12,300	1.24


Instructions Executed
Sampling Data (Not Issued)
Source
Sampling Data (All)
Instructions Executed
Total Sample Count: 111
Barrier: 43 (38.7%)
Mio Throttle: 21 (18.9%)
Not Selected: 8 (7.2%)
Selected: 7 (6.3%)
Short Scoreboard: 16 (14.4%)
Wait: 16 (14.4%)
Source/PTX/SASS analysis and correlation
Source metrics per instruction
Metric heatmap to quickly identify hotspots

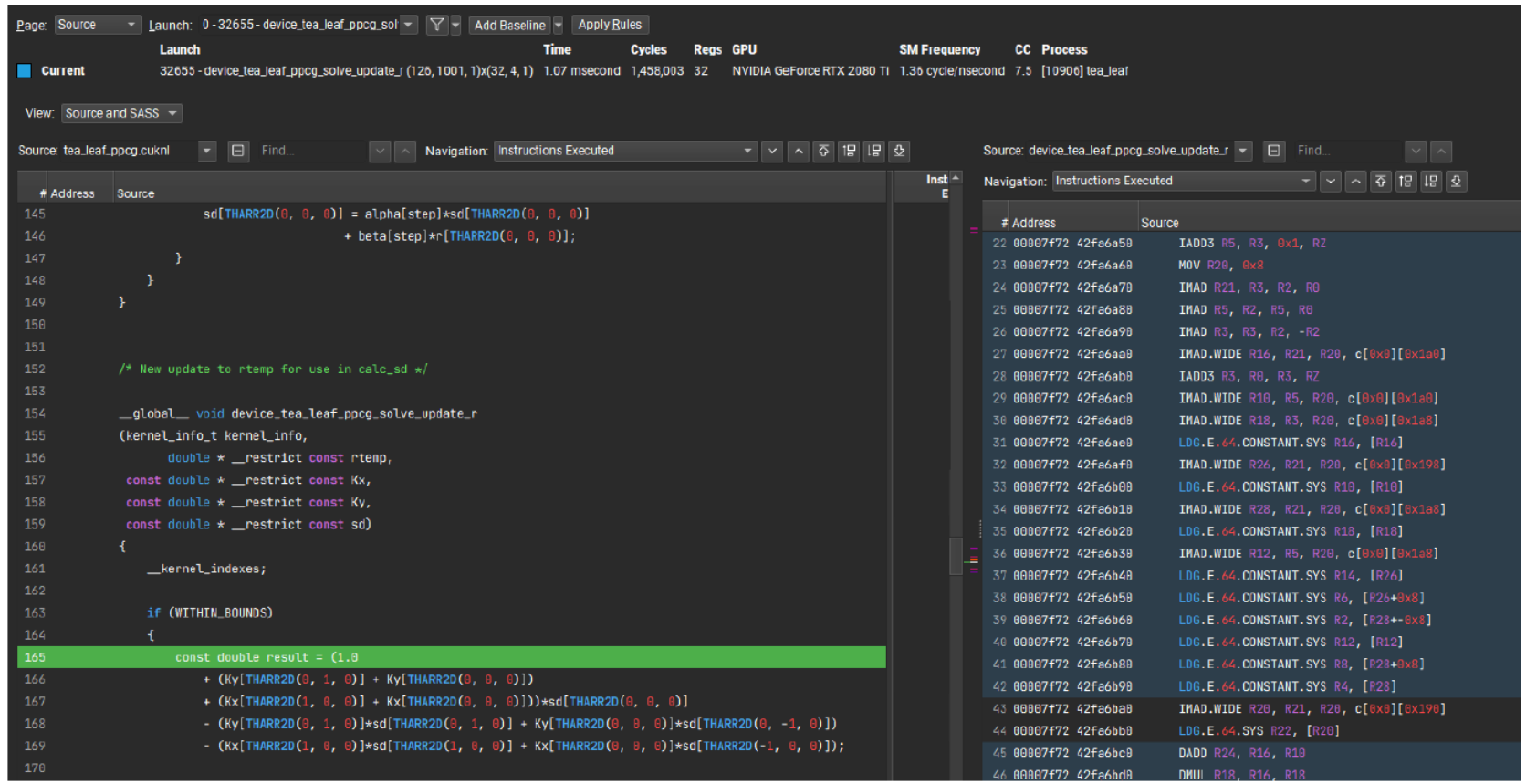
KERNEL PROFILES WITH NSIGHT COMPUTE

```
$ ncu –k mykernel –o report ./myapp.exe
```

- Generated file: report.ncu-rep
  - Open for viewing in the Nsight Compute UI
- (Without the –k option, Nsight Compute with profile everything and take a long time)

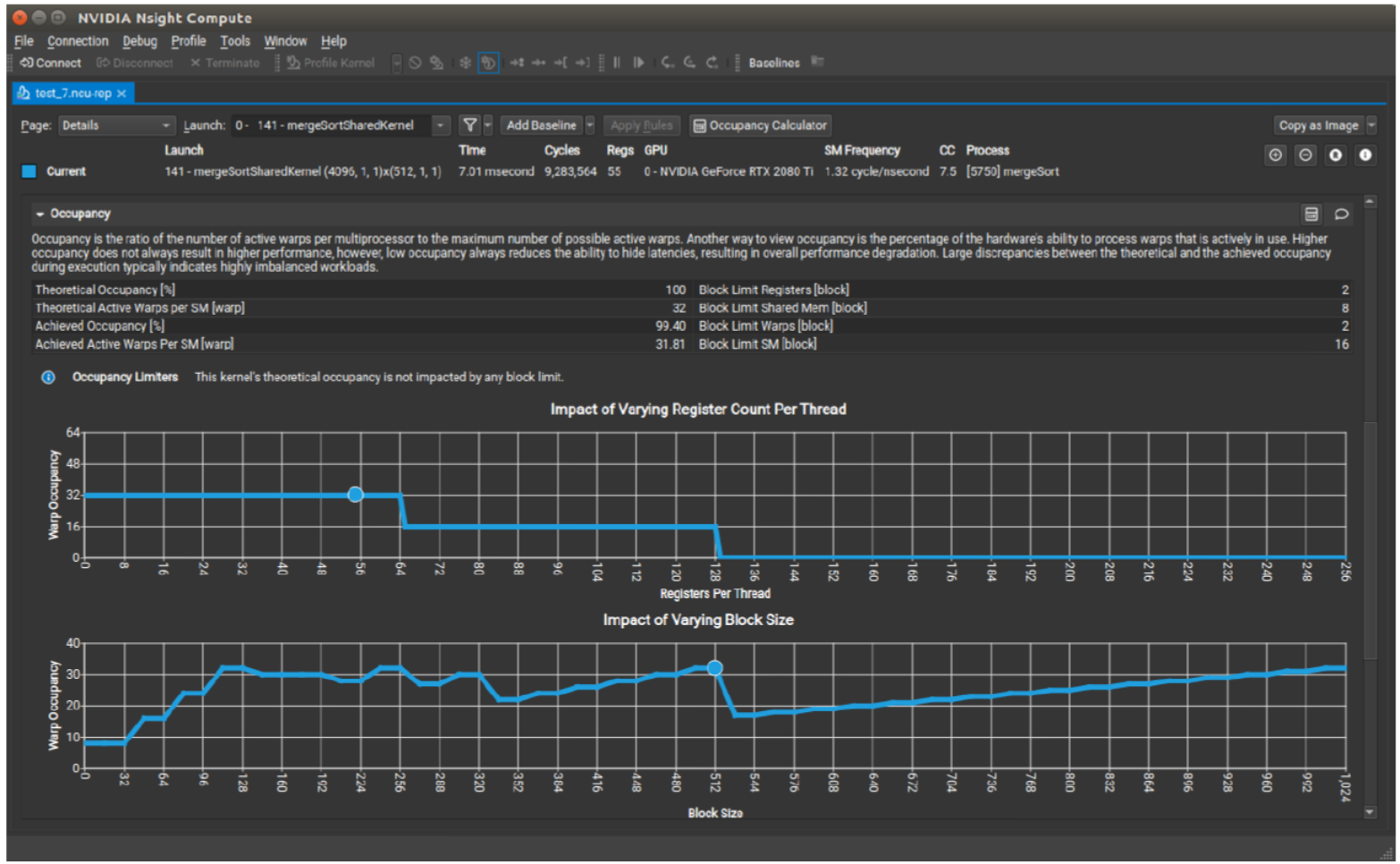
STANDALONE SOURCE VIEWER

- View of side-by-side assembly and correlated source code for CUDA kernels
- No profile required
- Open .cubin files directly
- Helps identify compiler optimizations and inefficiencies



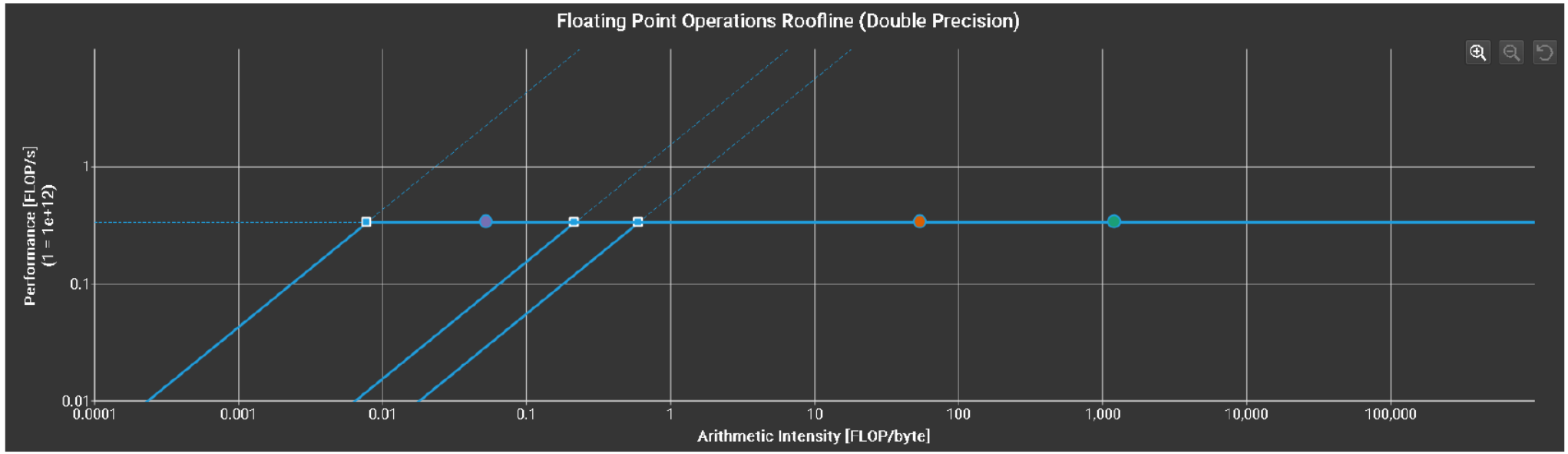
OCCUPANCY CALCULATOR

MODEL HARDWARE USAGE AND IDENTIFY LIMITERS



- Model theoretical hardware usage
- Understand limitations from hardware vs. kernel parameters
- Configure model to vary HW and kernel parameters
- Opened from an existing report or as a new activity

HIERARCHICAL ROOFLINE



- Visualize multiple levels of the memory hierarchy
- Identify bottlenecks caused by memory limitations
- Determine how modifying algorithms may (or may not) impact performance

Sections/Rules Info

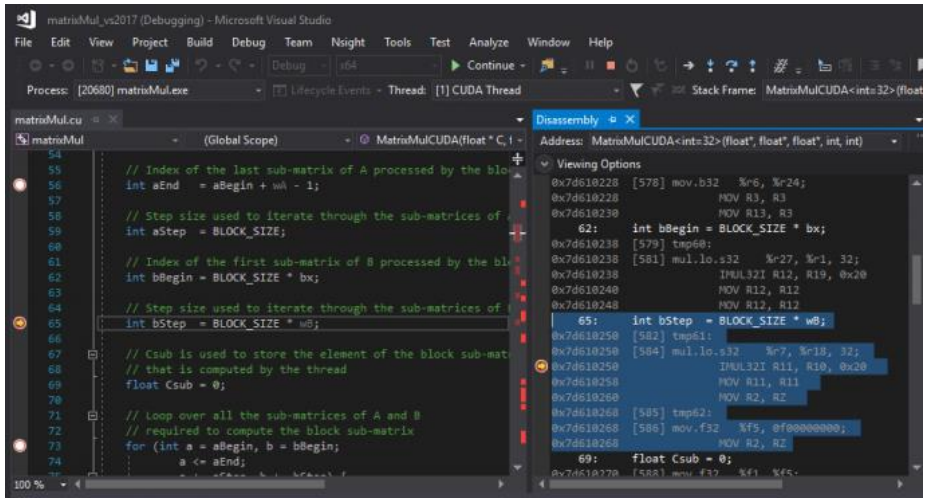
Sections/Rules | Reload | Enable All | Disable All | Restore

Enter filter

	Name	Priority	Description
☑	GPU Speed Of Light Throughput (1)	10	High-level overview of the throughput for compu...
☑	GPU Speed Of Light Roofline Chart (1)	11	High-level overview of the utilization for comput...
☑	GPU Speed Of Light Hierarchical Roofline Chart (Double Precision)	12	High-level overview of the utilization for comp...
☑	GPU Speed Of Light Hierarchical Roofline Chart (Half Precision)	12	High-level overview of the utilization for comput...
☑	GPU Speed Of Light Hierarchical Roofline Chart (Single Precision)	12	High-level overview of the utilization for comput...
☑	GPU Speed Of Light Hierarchical Roofline Chart (Tensor Core)	12	High-level overview of the utilization for comput...
☐	Compute Workload Analysis (2)	20	Detailed analysis of the compute resources of t...

DEVELOPER TOOLS

Debuggers: cuda-gdb, Nsight Visual Studio Edition

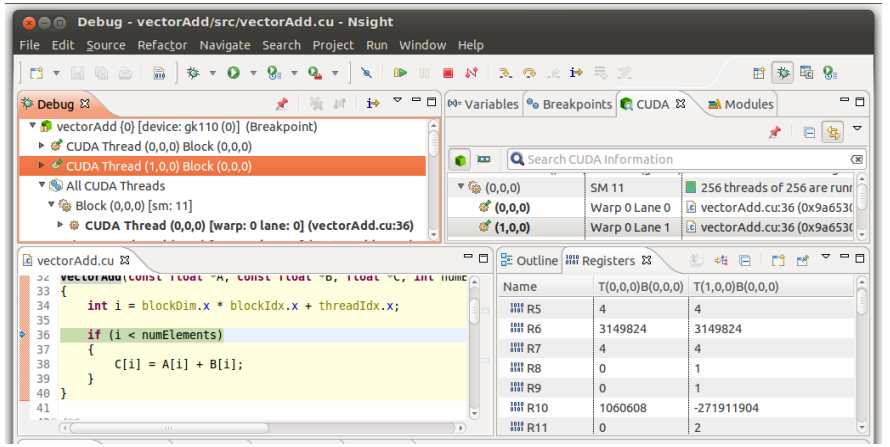


Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)

Correctness Checker:: Compute Sanitizer

```
$ compute-sanitizer --leak-check full memcheck_demo
========= COMPUTE-SANITIZER
Mallocing memory
Running unaligned_kernel
Ran unaligned_kernel: no error
Sync: no error
Running out_of_bounds_kernel
Ran out_of_bounds_kernel: no error
Sync: no error
========= Invalid __global__ write of size 4 bytes
=========     at 0x60 in memcheck_demo.cu:6:unaligned_kernel(void)
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x400100001 is misaligned
```

IDE integrations: Nsight Eclipse Edition
Nsight Visual Studio Edition
Nsight Visual Studio **Code** Edition



NVIDIA

CUDA GDB

COMMAND LINE AND IDE BACKEND DEBUGGER

- Unified CPU and CUDA Debugging
- CUDA-C/PTX/SASS support
- Built on GDB and uses many of the same CLI commands

```
(cuda-gdb) info cuda threads breakpoint all
  BlockIdx ThreadIdx         Virtual PC Dev SM Wp Ln         Filename  Line
Kernel 0
   (1,0,0)   (0,0,0) 0x0000000000948e58   0 11  0  0 infoCommands.cu    12
   (1,0,0)   (1,0,0) 0x0000000000948e58   0 11  0  1 infoCommands.cu    12
   (1,0,0)   (2,0,0) 0x0000000000948e58   0 11  0  2 infoCommands.cu    12
   (1,0,0)   (3,0,0) 0x0000000000948e58   0 11  0  3 infoCommands.cu    12
   (1,0,0)   (4,0,0) 0x0000000000948e58   0 11  0  4 infoCommands.cu    12
   (1,0,0)   (5,0,0) 0x0000000000948e58   0 11  0  5 infoCommands.cu    12

(cuda-gdb) info cuda threads breakpoint 2 lane 1
  BlockIdx ThreadIdx         Virtual PC Dev SM Wp Ln         Filename  Line
Kernel 0
   (1,0,0)   (1,0,0) 0x0000000000948e58   0 11  0  1 infoCommands.cu    12
```

COMPUTE SANITIZER

AUTOMATICALLY SCAN FOR BUGS AND MEMORY ISSUES

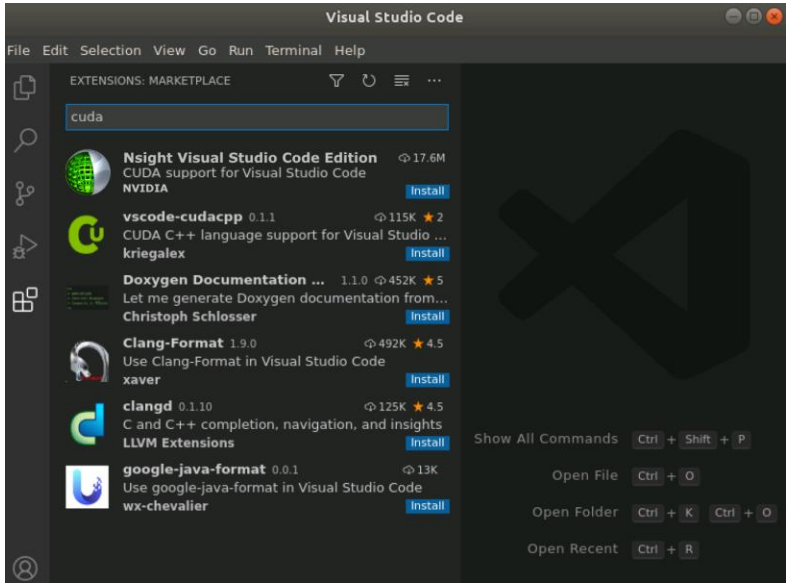
- Compute sanitizer checks correctness issues via sub-tools:
  - *Memcheck* – Memory access error and leak detection
  - *Racecheck* – Shared memory data access hazard detection
  - *Initcheck* – Uninitialized device global memory access detection
  - *Synccheck* – Thread synchronization hazard detection

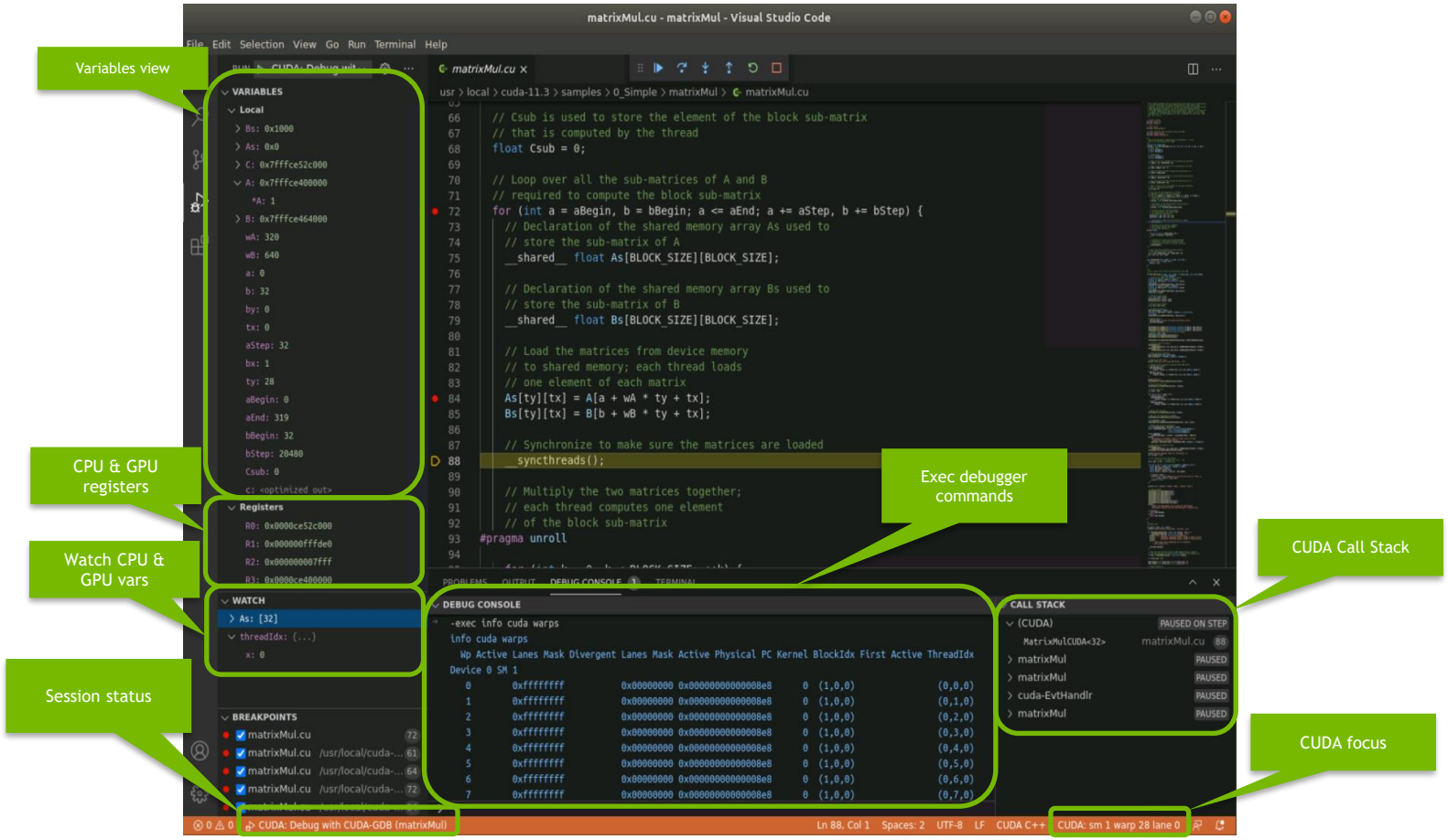
```
$ compute-sanitizer --leak-check full memcheck_demo
========= COMPUTE-SANITIZER
Mallocing memory
Running unaligned_kernel
Ran unaligned_kernel: no error
========= Invalid __global__ write of size 4 bytes
=========     at 0x160 in memcheck_demo.cu:6:unaligned_kernel()
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x7fa73cc00001 is misaligned
=========     Saved host backtrace up to driver entry point at kernel launch time
=========     Host Frame: [0x2068ea]
=========                 in /usr/lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame: [0xfc4b]
=========                 in memcheck_demo
=========     Host Frame: [0x66288]
=========                 in memcheck_demo
=========     Host Frame: [0xb4eb]
=========                 in memcheck_demo
=========     Host Frame:__device_stub__Z16unaligned_kernelv(void) [0xb1fc]
=========                 in memcheck_demo
=========     Host Frame:unaligned_kernel(void) [0xb257]
=========                 in memcheck_demo
=========     Host Frame:run_unaligned(void) [0xaf38]
=========                 in memcheck_demo
=========     Host Frame:main [0xb0b1]
=========                 in memcheck_demo
=========     Host Frame:__libc_start_main [0x28cb2]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:_start [0xad9e]
=========                 in memcheck_demo
```

NSIGHT VISUAL STUDIO CODE EDITION

Visual Studio Code extensions that provides:

- CUDA code syntax highlighting
- CUDA code completion
- Build warning/errors
- Debug CPU & GPU code
- Remote connection support via SSH
- Available on the VS Code Marketplace now!





https://developer.nvidia.com/nsight-visual-studio-code-edition

NVIDIA

OPEN HACKATHONS powered by OpenACC

Home | Events | Attendees | Mentors | About | OpenACC

NASA GPU Hackathon 2022

Date(s): Sep 19, 2022 - Sep 28, 2022
Event Focus: **HPC+AI**
North America/Latin America

Applications Closed

CSCS GPU Hackathon 2022

Date(s): Sep 19, 2022 - Sep 29, 2022
Event Focus: **HPC+AI**
Europe/Middle East/Africa

Applications Closed

NVIDIA/HLRS SciML GPU Bootcamp

Date(s): Oct 24, 2022 - Oct 25, 2022
Event Focus: **HPC**
Europe/Middle East/Africa

Applications Open

OLCF GPU Hackathon 2022

Date(s): Oct 17, 2022 - Oct 27, 2022
Event Focus: **HPC+AI**
North America/Latin America

Applications Open

NERSC GPU Hackathon 2022

Date(s): Nov 30, 2022 - Dec 8, 2022
Event Focus: **HPC+AI**
North America/Latin America

Applications Open

USEFUL LINKS

Web: https://developer.nvidia.com/tools-overview

How to contact us?

Forums: https://forums.developer.nvidia.com/c/development-tools

email: devtools-support@nvidia.com

Other digital GTC talks of interest:

S21351: Scaling the Transformer Model Implementation in PyTorch Across Multiple Nodes

S21547: Rebalancing the Load:Profile-Guided Optimization of the NAMD Molecular Dynamics Program for Modern GPUs using Nsight Systems

S21771: Optimizing CUDA Kernels in HPC Simulation and Visualization Codes using Nsight Compute

S21565: Roofline Performance Model for HPC and Deep-Learning Applications