NVIDIA

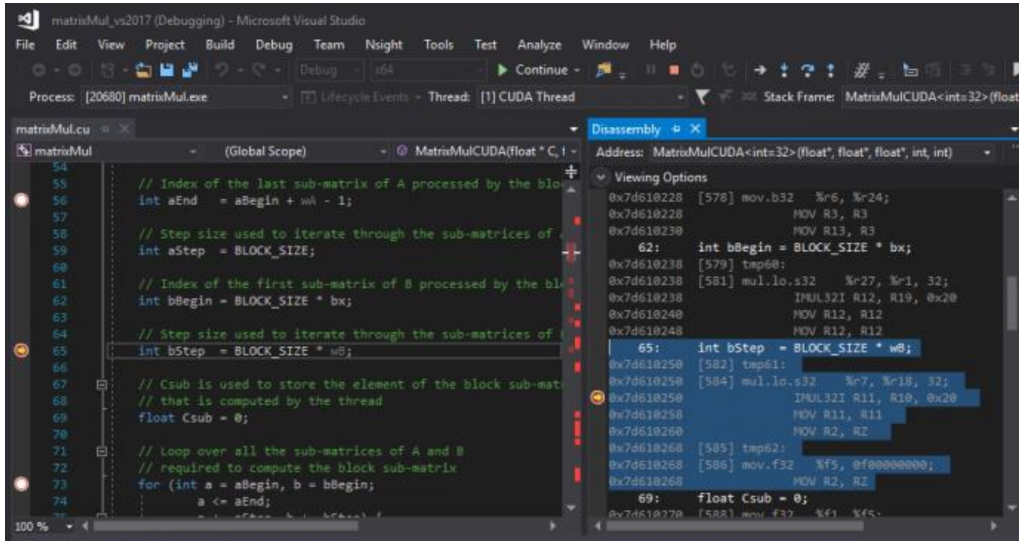
NVIDIA DEVELOPER TOOLS

Robert Searles

Senior Solutions Architect, NVIDIA

Developer Tools Ecosystem

Debuggers: cuda-gdb, Nsight Visual Studio Edition Nsight Visual Studio **Code** Edition

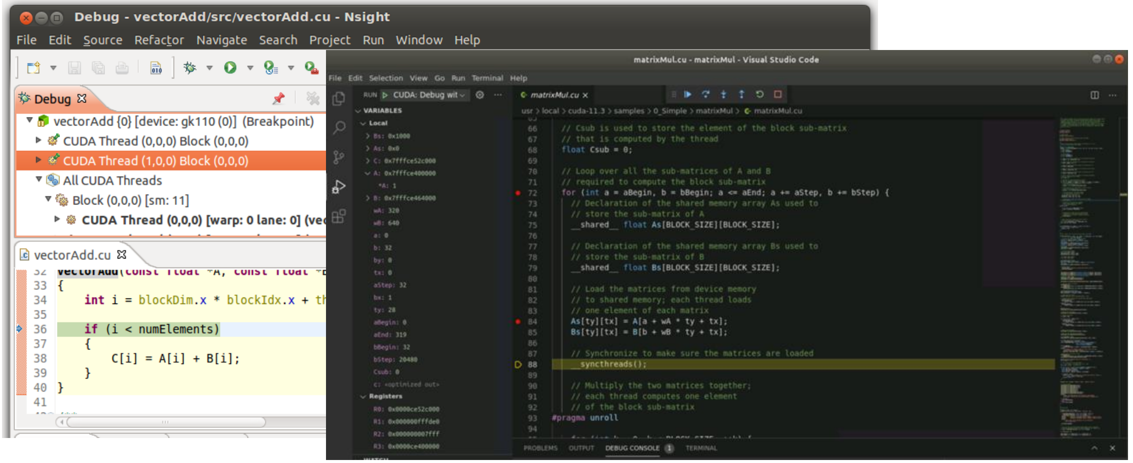


Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)

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 Visual Studio **Code** Edition, Nsight Visual Studio Edition, Nsight Eclipse Edition



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, ...);
```

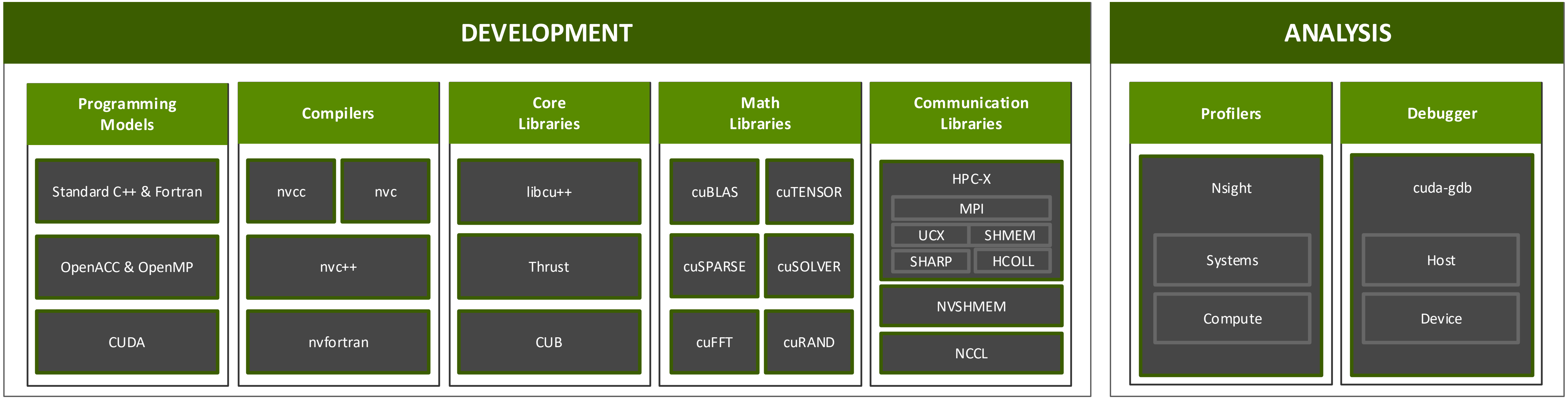
ACCELERATION LIBRARIES

Core	Math	Communication	Data Analytics	AI	Quantum

NVIDIA

NVIDIA HPC SDK

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

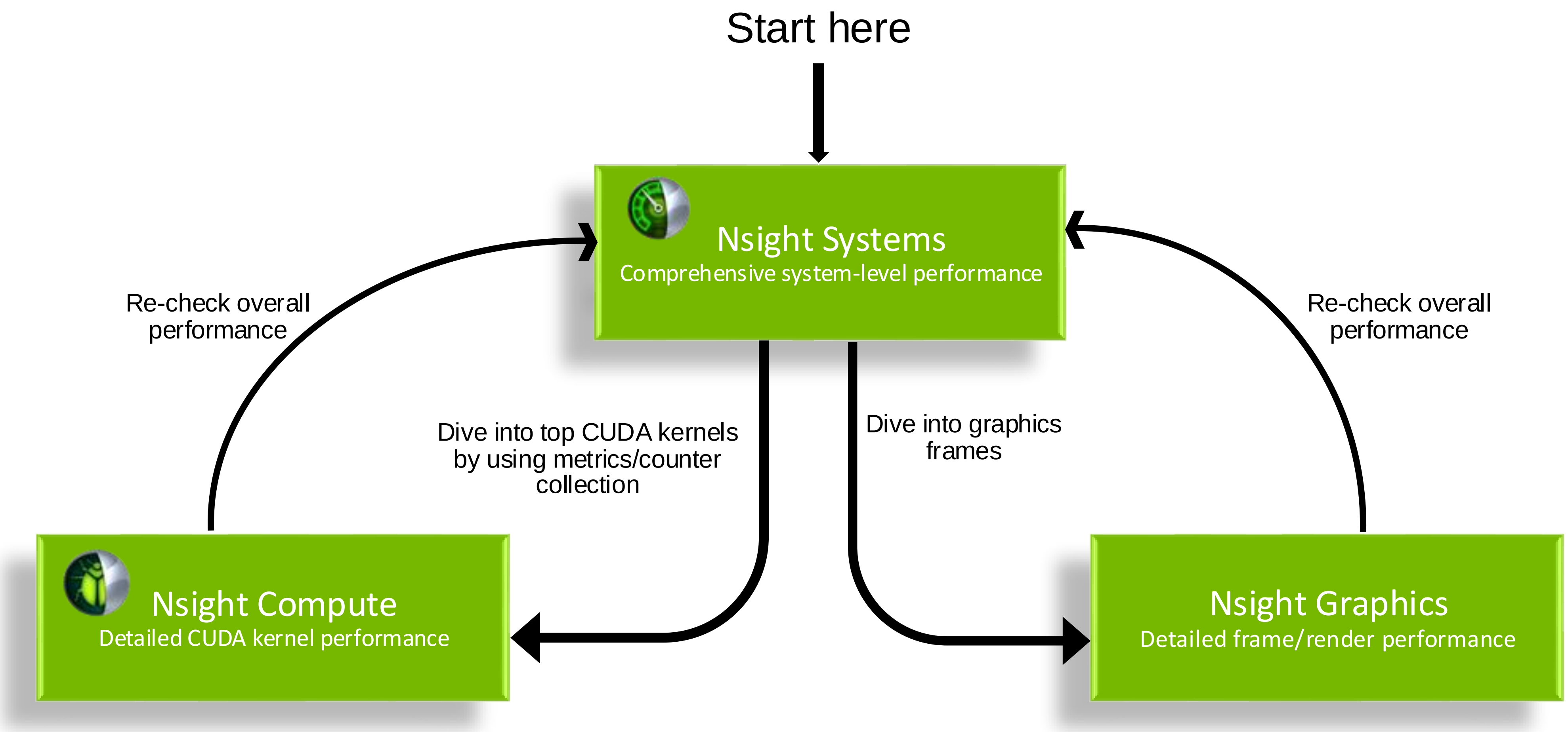


Develop for the NVIDIA Platform: GPU, CPU and Interconnect

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

7-8 Releases Per Year | Freely Available

NSIGHT Profilers Workflow

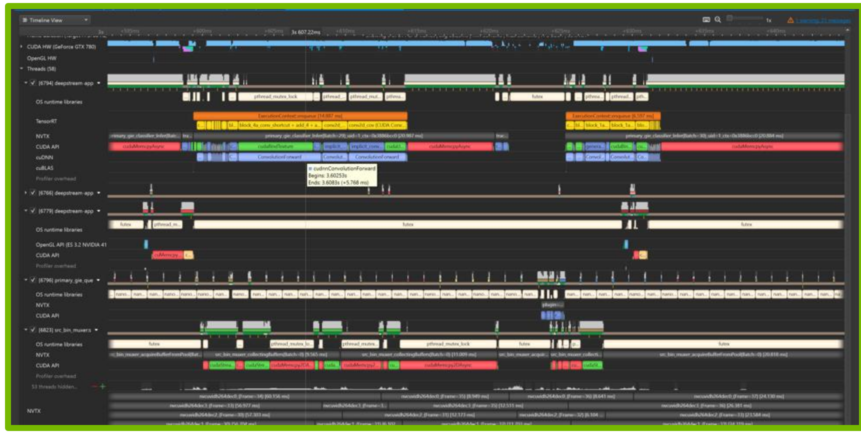


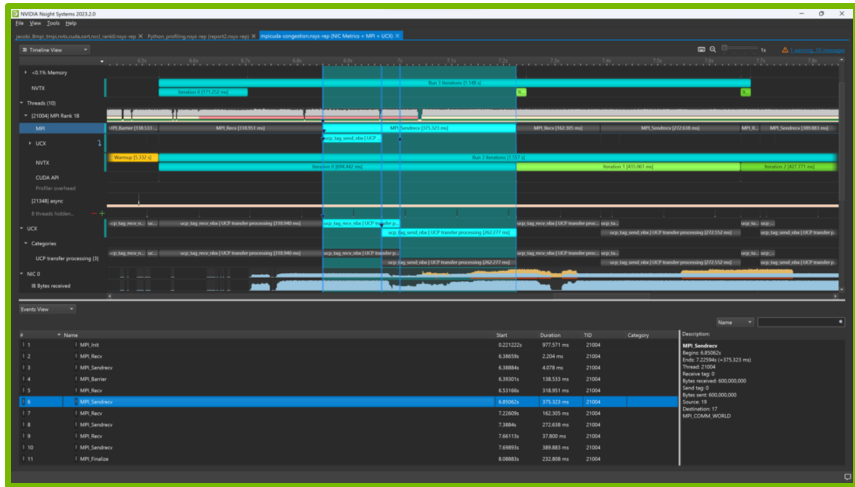
Nsight Systems

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
  - Identify 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, ARM Server, Tegra), Windows, macOS X (host)
- GPUs: Pascal+
- Docs/product: https://developer.nvidia.com/nsight-systems





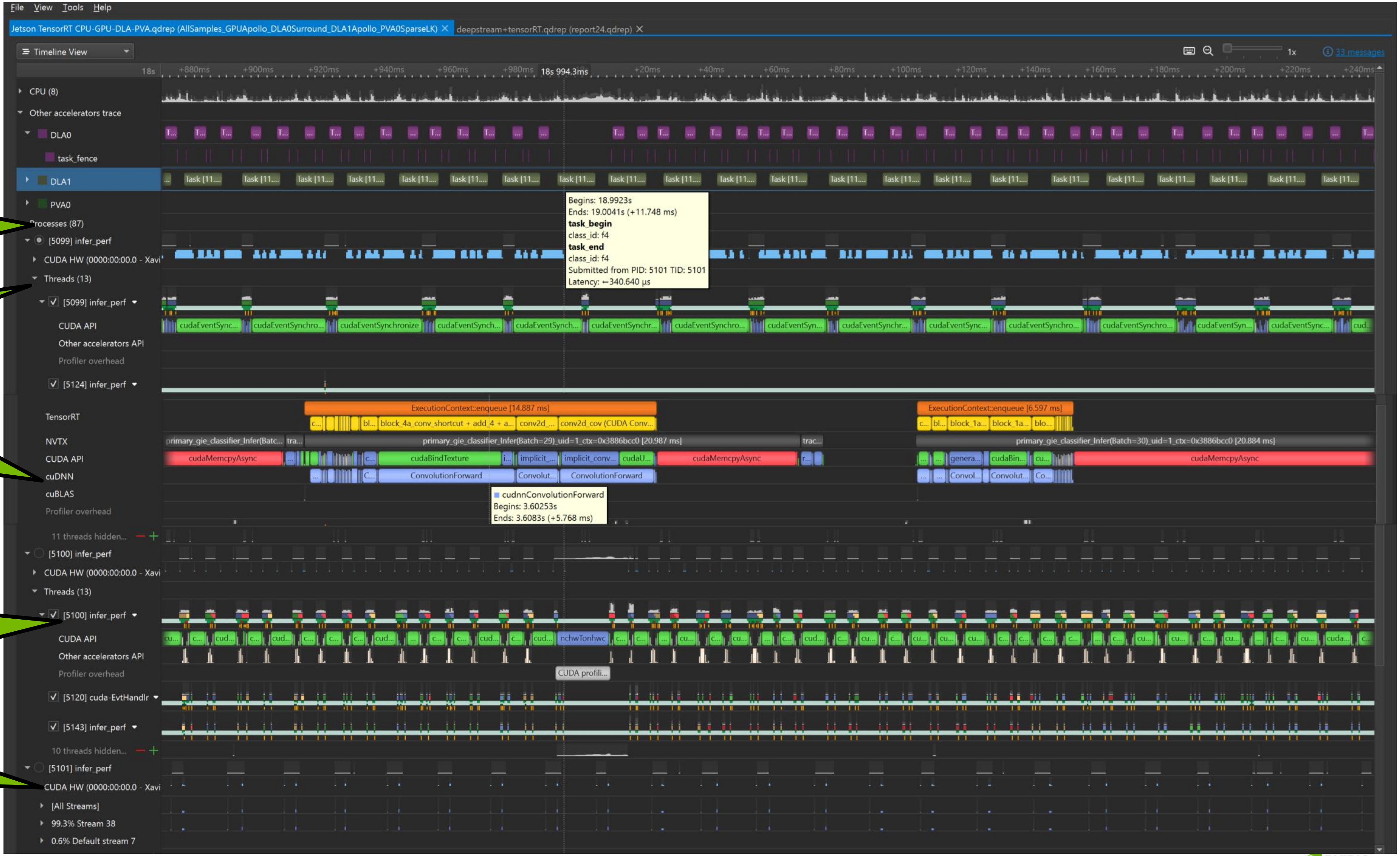
Processes and threads

Thread state

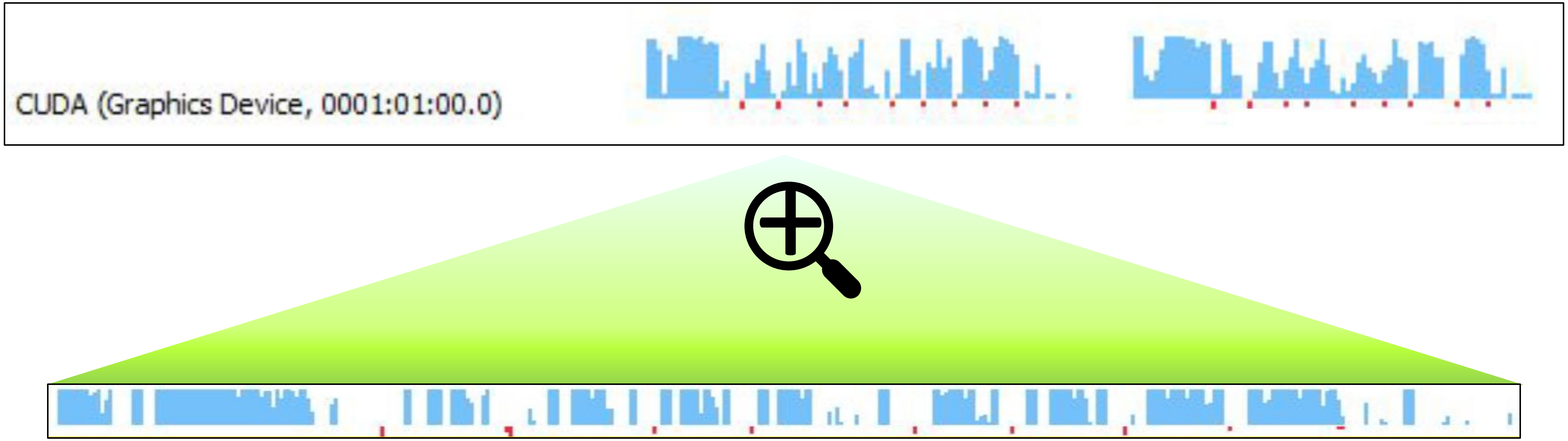
cuDNN and cuBLAS trace

Kernel and memory transfer activities

Multi-GPU

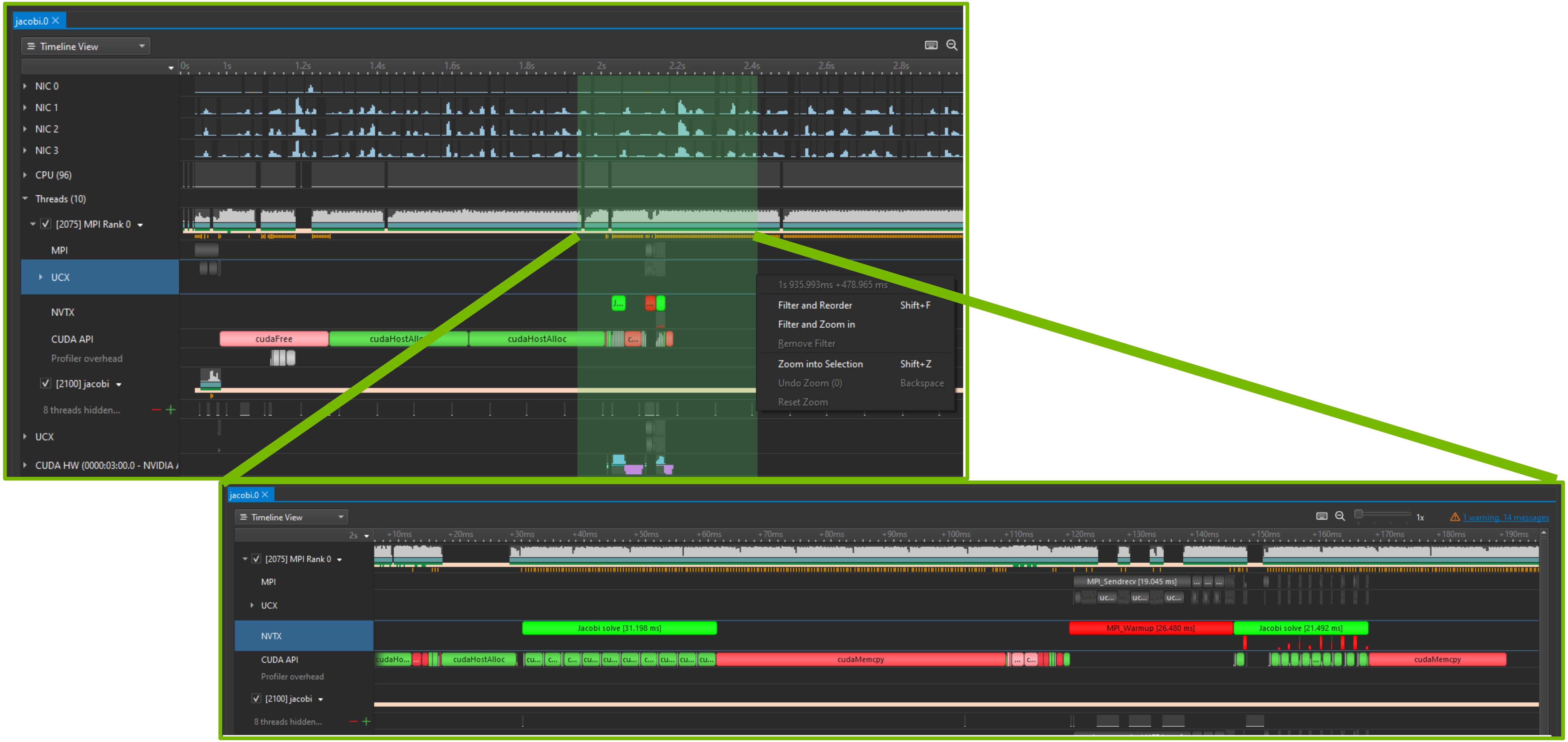


Reading Utilization



- Zoom in to valleys to find gaps!

Zoom/Filter to Exact Areas of Interest

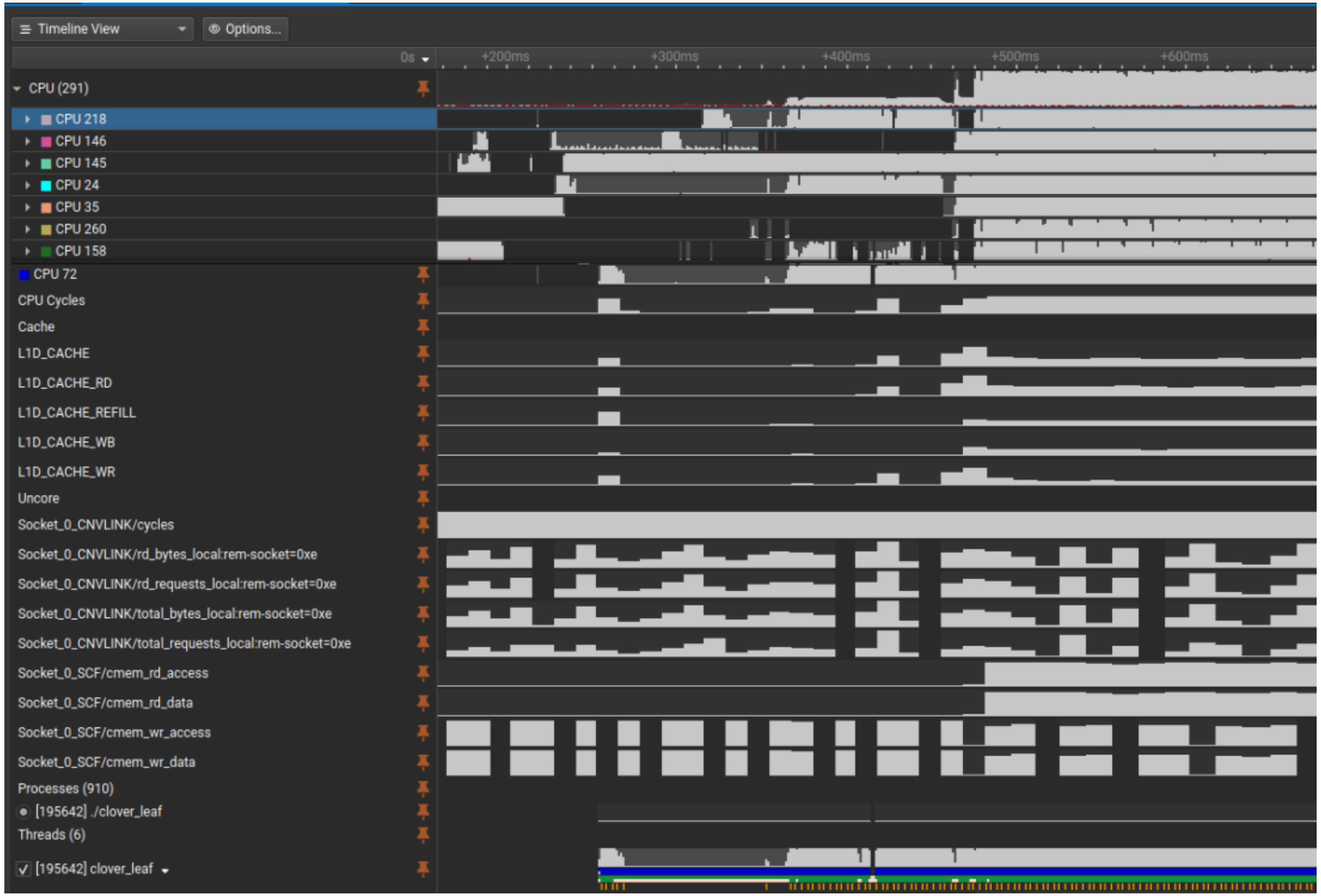


NVIDIA

Grace Host Profiling

Hardware Counters and Metrics

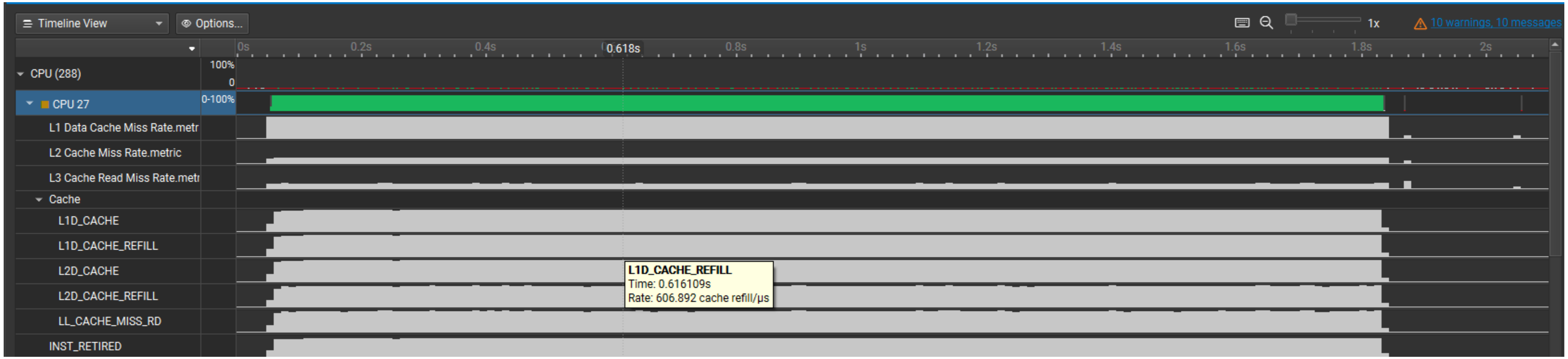
- CPU Core and Uncore Events
  - Sampled for each CPU
  - Visualize parallelism effects
  - Cache hit/miss, instructions retired, etc…
  - L3 Coherency Fabric
  - Socket to socket traffic
- Variable sampling frequencies supported
- Timeline correlated with all other data
  - GPU vs. CPU idle times and metrics
  - Data movement
  - Zoom and filter



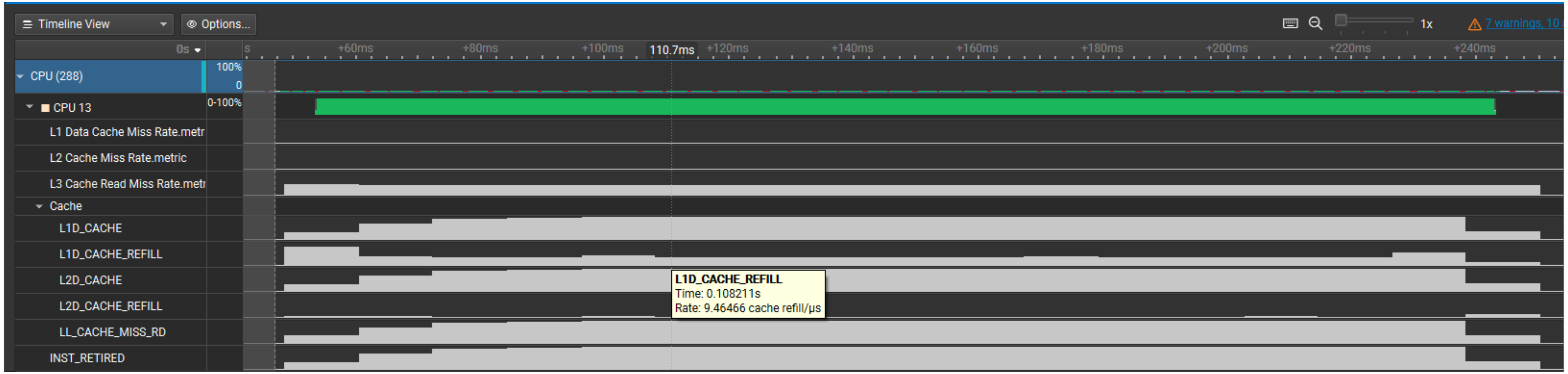
Grace Host Profiling

Cache Access Pattern Example

Single threaded CPU matrix multiplication with poor memory access patterns

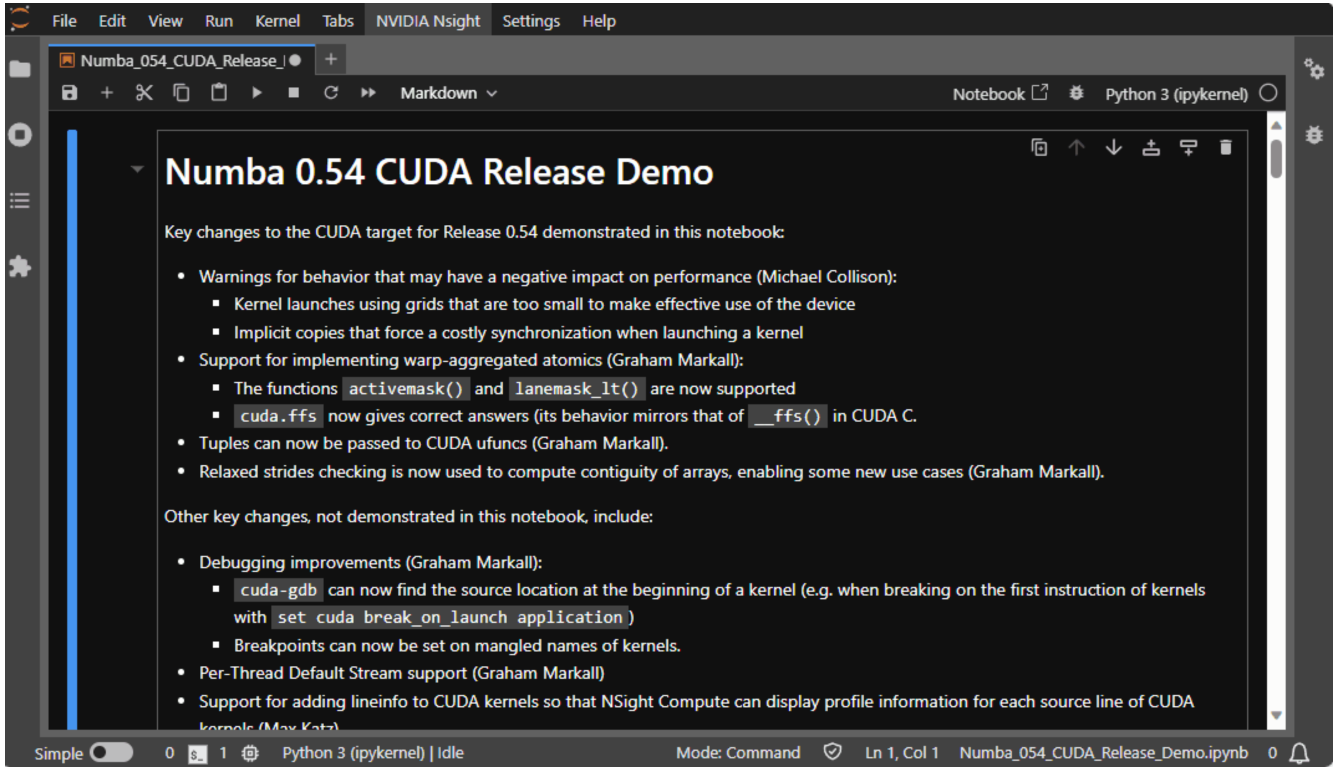


Improving access pattern and implementing cache blocking



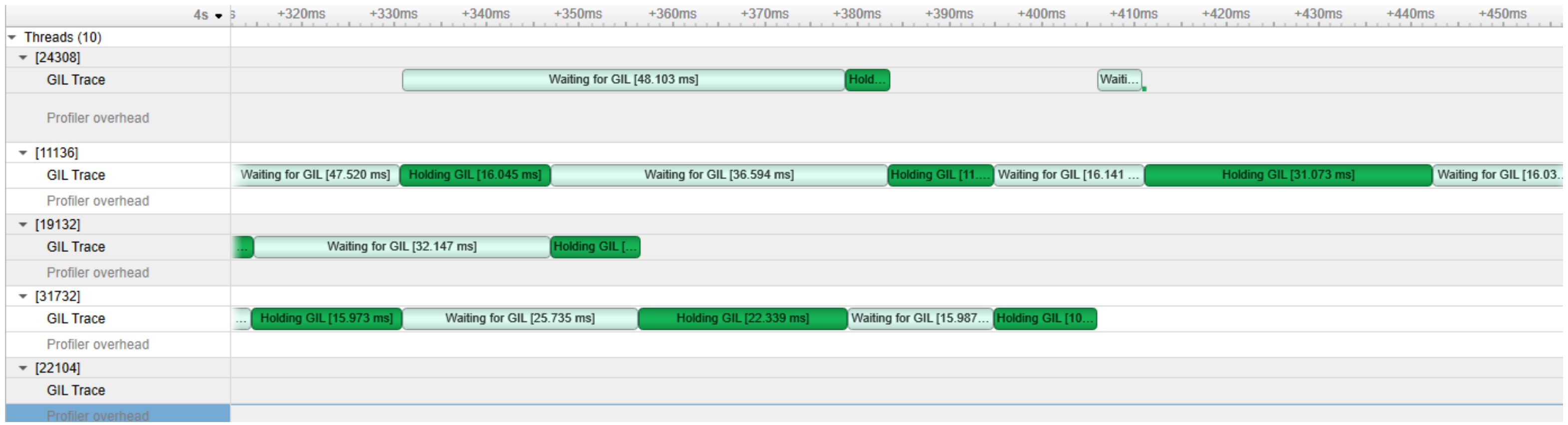
JupyterLab Integration Updates

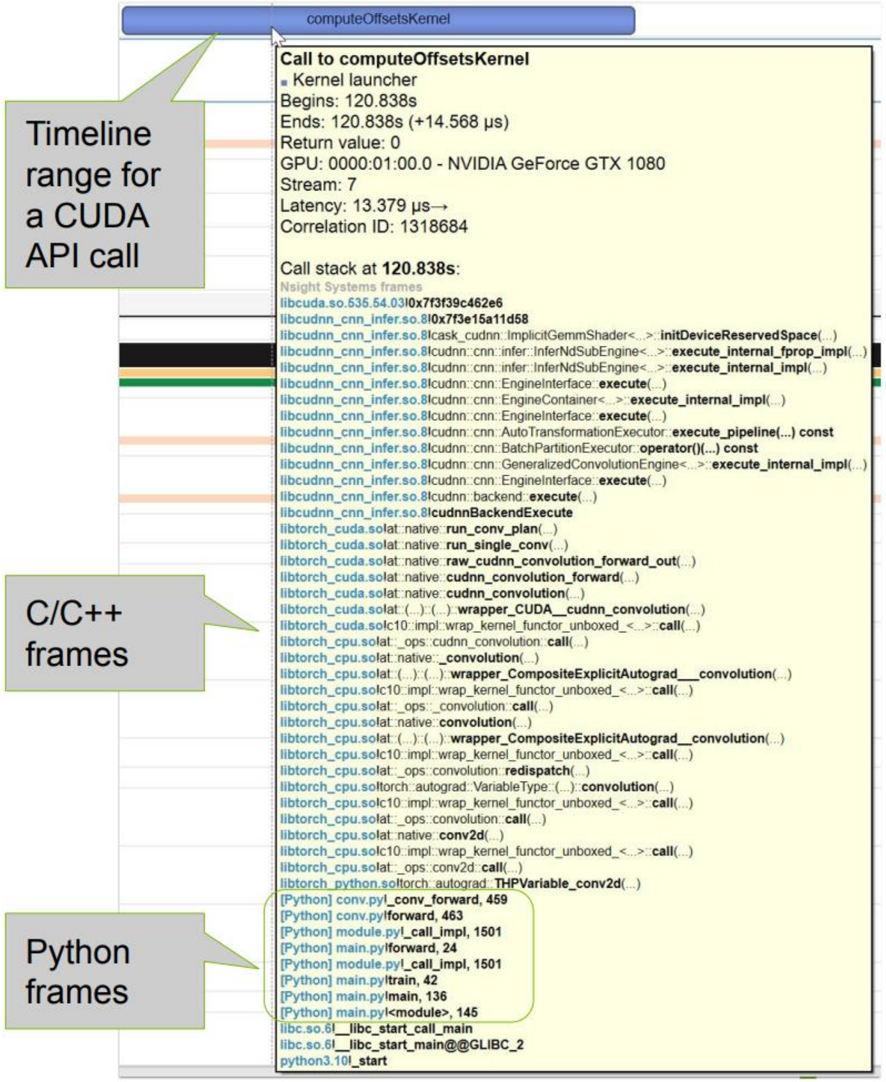
- Extension to JupyterLab
- Profile individual Jupyter cells
- Text-based results can be viewed directly in Jupyter
- Launch **new** remote GUI streaming container directly in JupyterLab
  - Servers without X, Windowing Manager, …
  - Container with X, WM, & WebRTC server
  - Dockerfile inside Nsight Systems Installer
- See it in action:
  - DLIT61667: Profilers, Python, and Performance: Nsight Tools for Optimizing Modern CUDA Workloads



Python Profiling Updates

- Python Call Stacks Samples and CUDA API Backtrace
  - Identify where you are and how you got there
- Global Interpreter Lock (GIL) trace
  - Common performance limiter in Python
- See annotated code ranges built into in popular frameworks and libraries such as:
  - RAPIDS, Spark, CV-CUDA, and more…

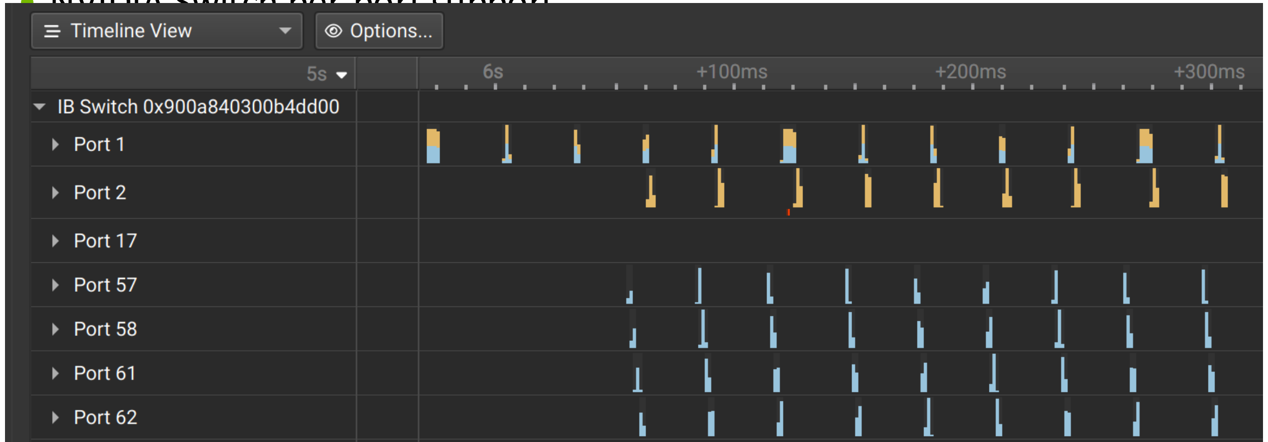


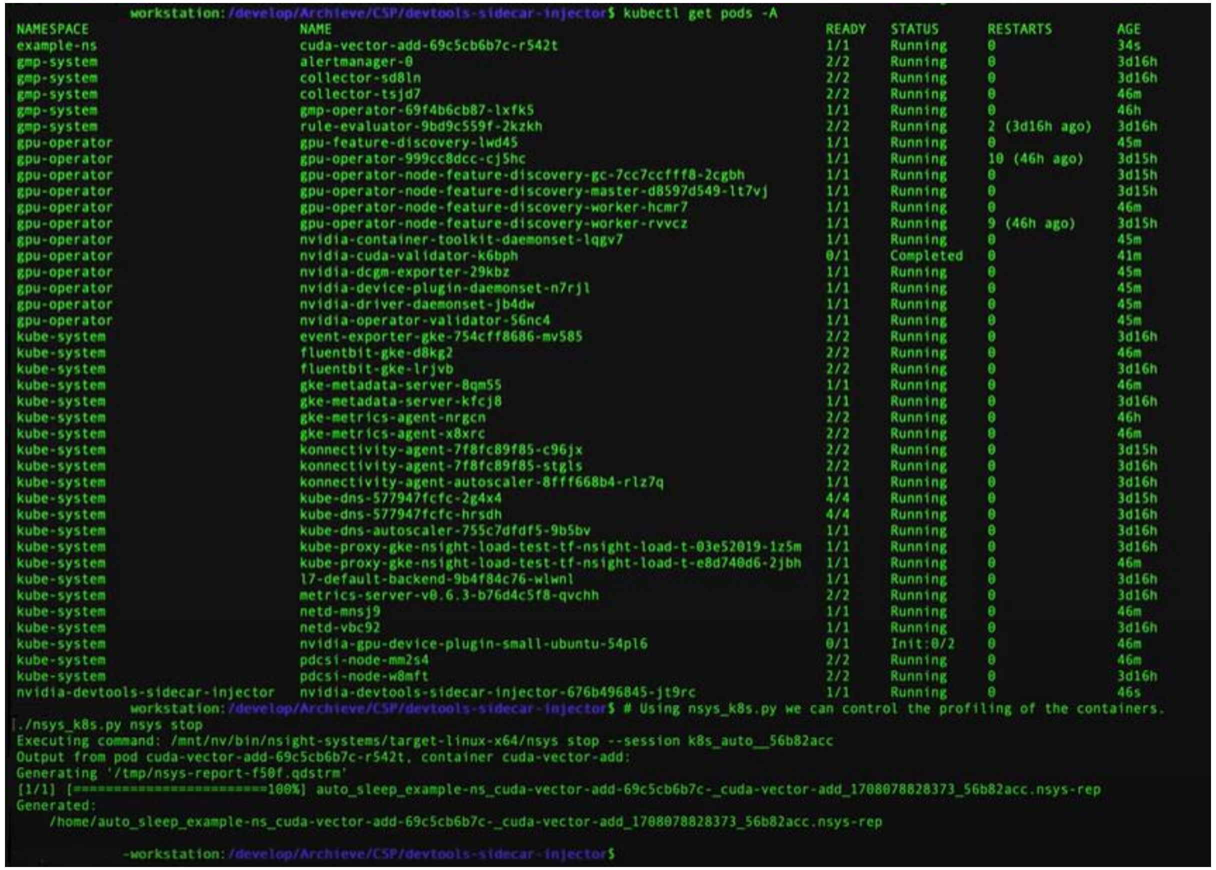


NVIDIA

Cluster and Recipe Framework Improvements

- Nsight Systems enhanced support for Kubernetes
- Nsight Systems analysis framework:
  - User programmable and predefined recipes to:
    - Process and analyze complex and large reports or collection of reports
    - Understand how compute cold-spots relate to communications
    - Generate multi-node heatmaps to show :
      - InfiniBand congestion
      - InfiniBand, Ethernet, and NVLink throughputs
      - Overlapped compute and networking
- NVIDIA Switch per port support





NVIDIA Tools eXtension (NVTX)

- Decorate application source code with annotations (markers, ranges, nested ranges, …) to help visualize execution with debugging, tracing and profiling tools
- Header-only library https://github.com/NVIDIA/NVTX/tree/release-v3/c.

```
#include <nvtx3/nvToolsExt.h>
```

- Marker:

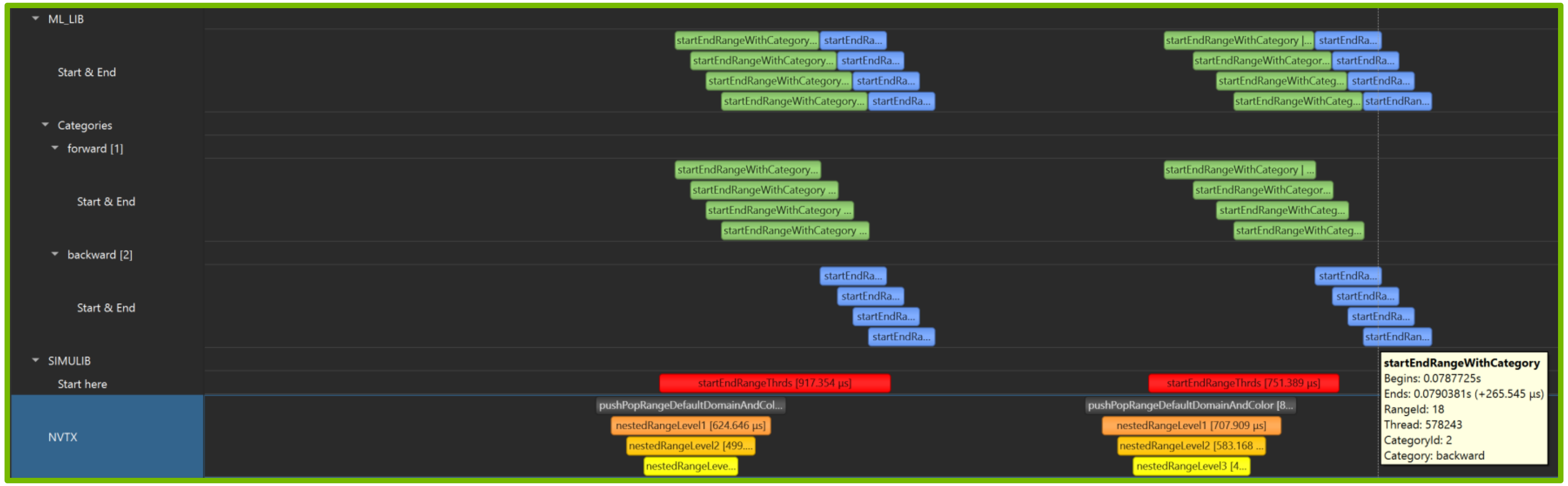
```
nvtxMark("This is a marker");
```

- Push-Pop range

```
nvtxRangePush("This is a push/pop range");
// Do something interesting in the range
nvtxRangePop();  // Pop must be on same thread as corresponding Push
```

- Start-End range

```
nvtxRangeHandle_t handle = nvtxRangeStart("This is a start/end range");
// Somewhere else in the code, not necessarily same thread as Start call:
nvtxRangeEnd(handle);
```



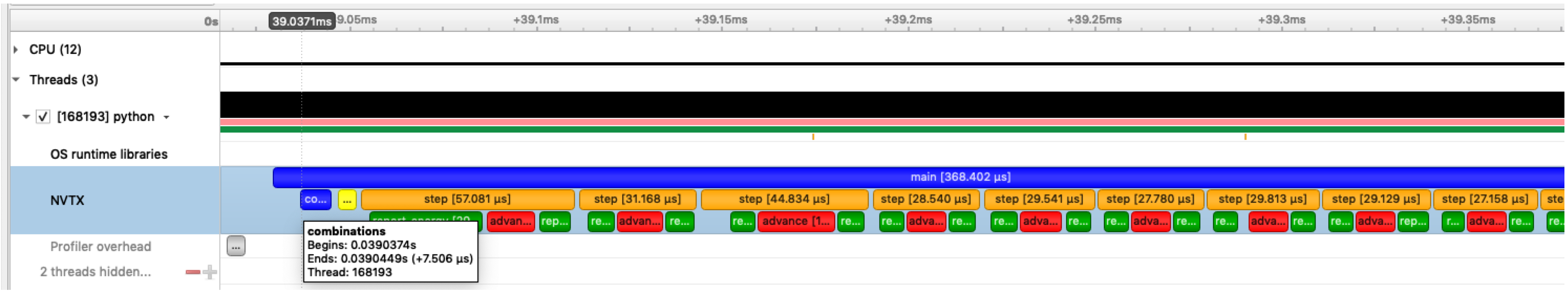
API references https://nvidia.github.io/NVTX/doxygen/index.html and https://nvidia.github.io/NVTX/doxygen-cpp/index.html

Python and NVTX

- Annotate Python code with NVTX

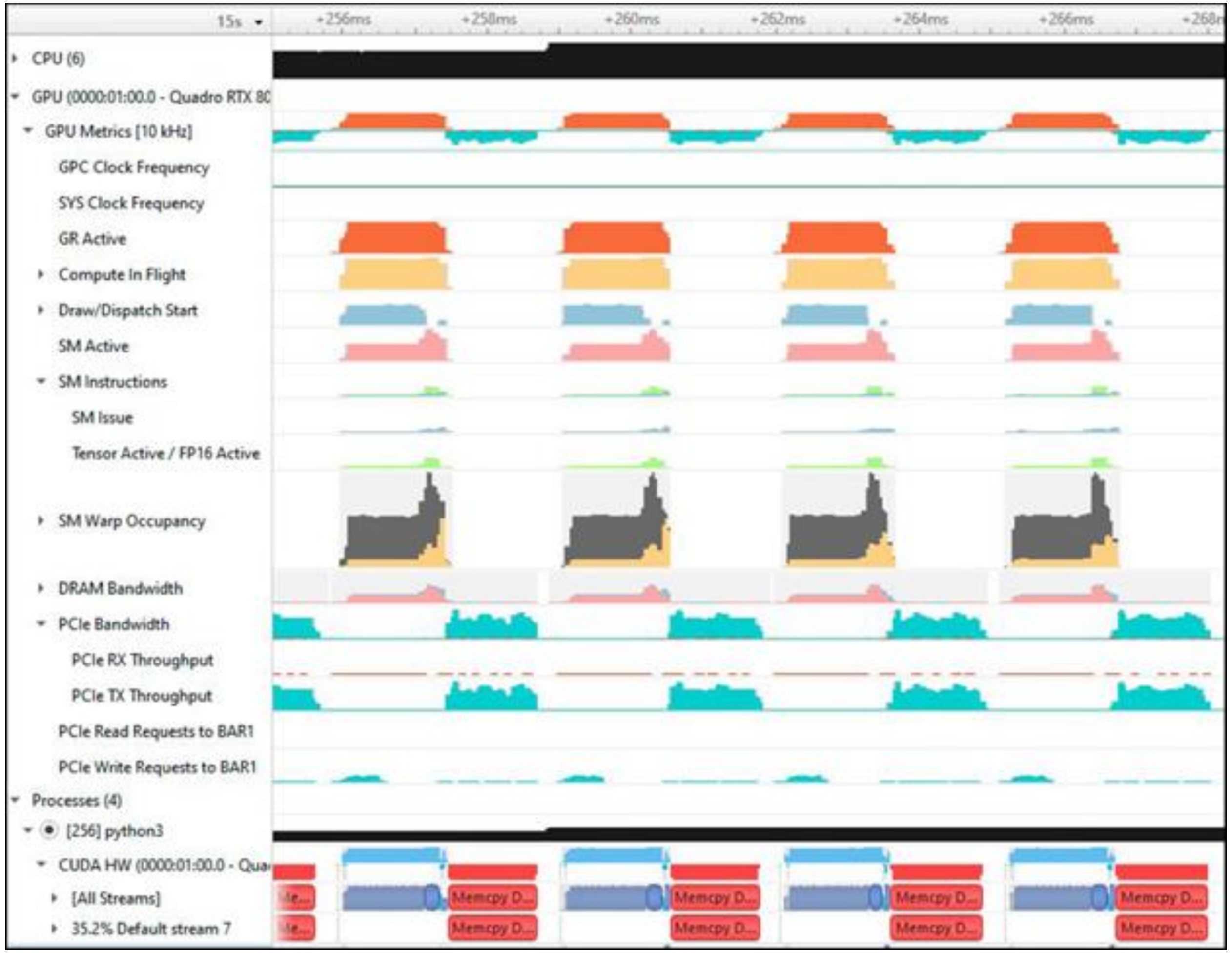
```
# demo.py

import time
import nvtx

@nvtx.annotate(color="blue")
def my_function():
    for i in range(5):
        with nvtx.annotate("my_loop", color="red"):
            time.sleep(i)

my_function()
```

- pip install nvtx - https://pypi.org/project/nvtx/



NVIDIA

GPU Metrics Sampling



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

NVIDIA

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 batch 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

General Optimization Tips

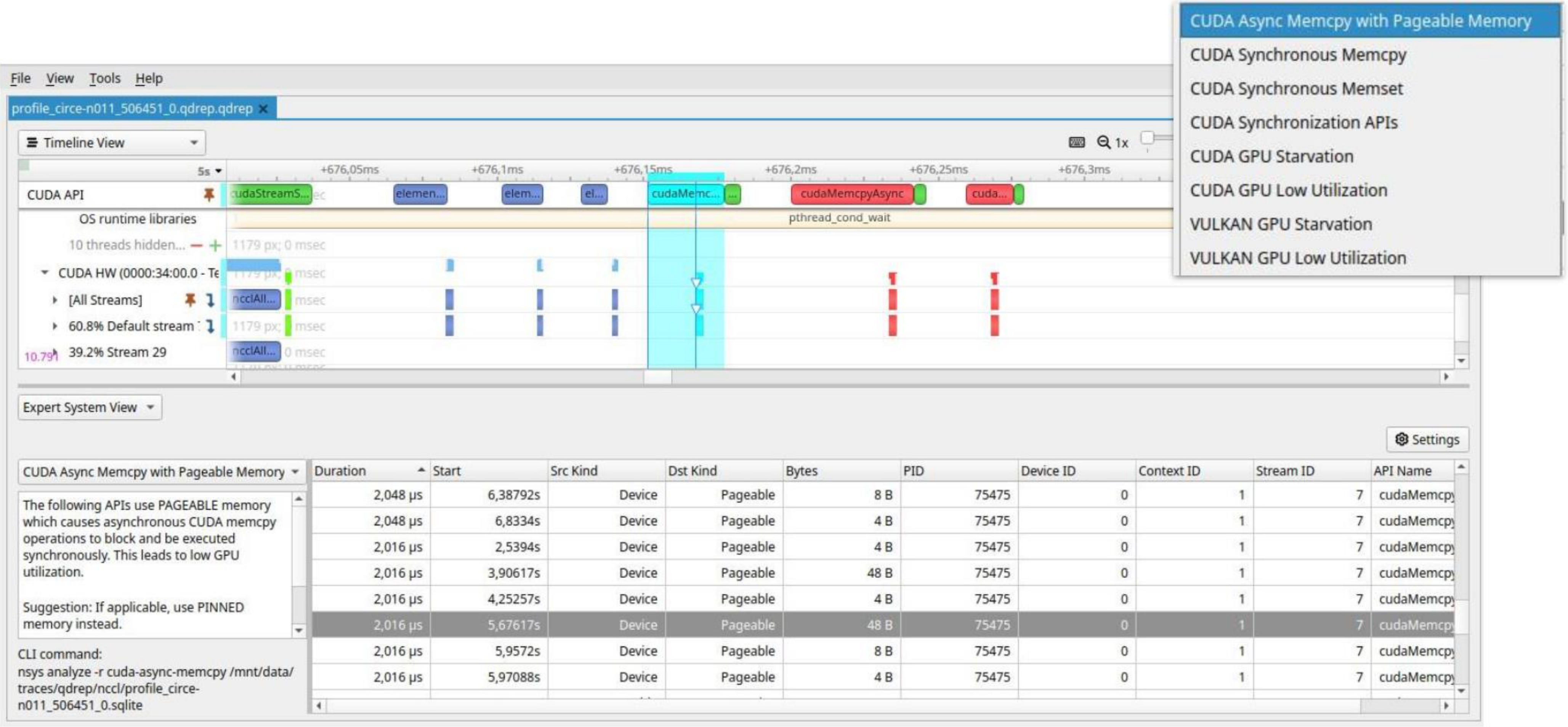
- Using tensor cores?
  - Minimize conversions/transposes
- Increase grid and batch size to utilize GPUs width
- Conventional parallelism – more worker threads!
- Parallel pipelining
  - No data dependency? Parallelize!
  - Prefetch next batch/iteration during computation
- Can I reorder sooner?

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

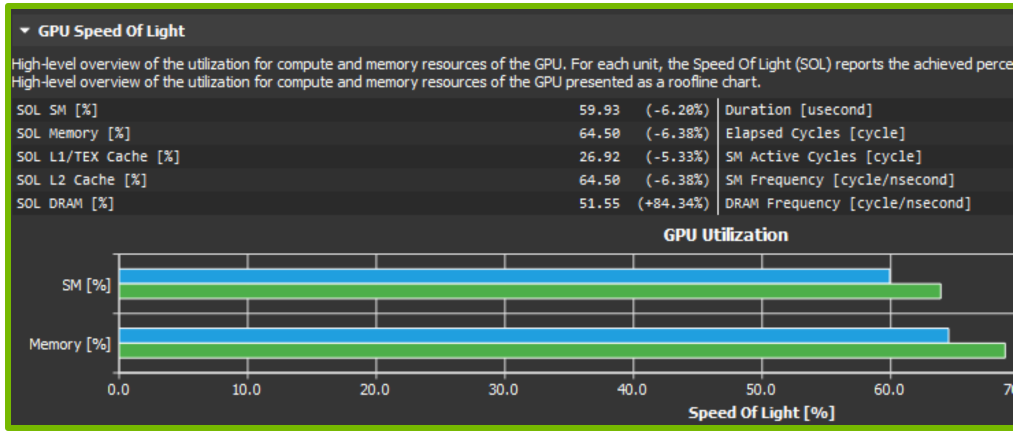
Kernel Profiler

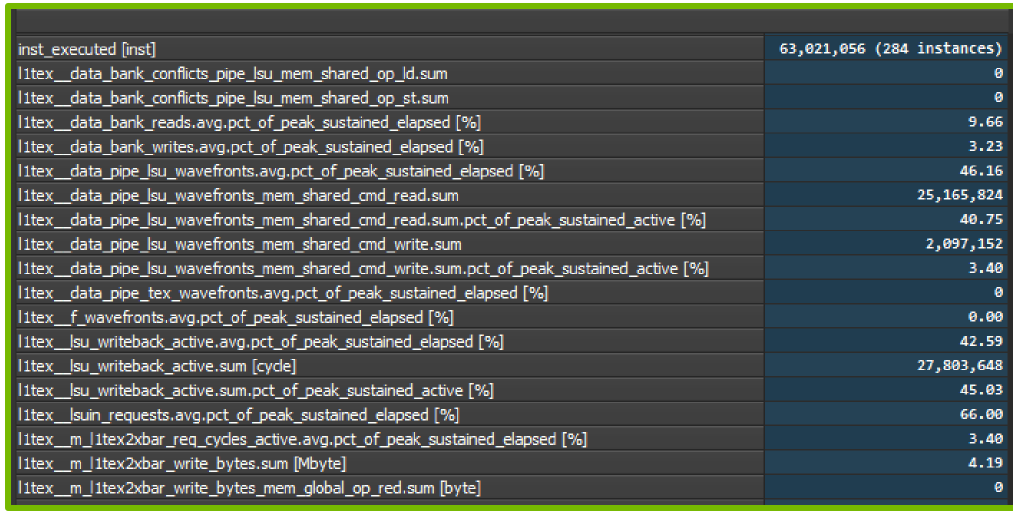
Key Features:

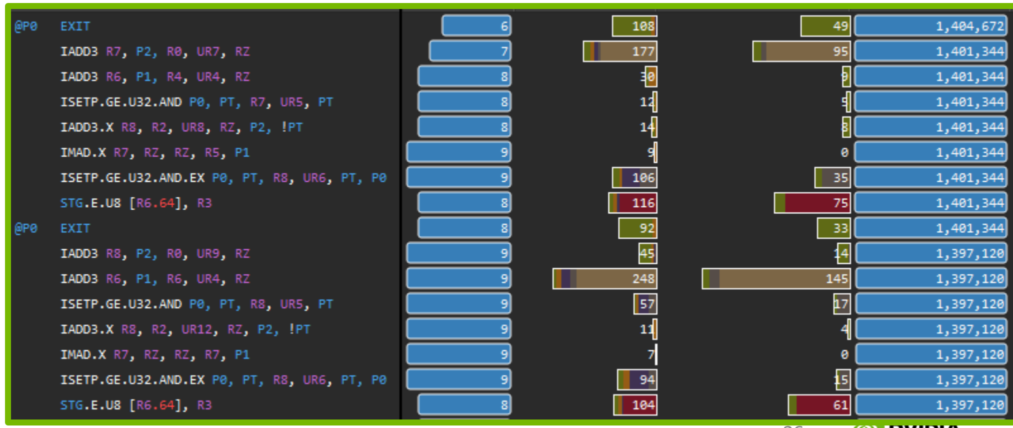
- Interactive CUDA API debugging and kernel profiling
- Built-in rules expertise
- Fully customizable data collection and display
- Command Line, Standalone, IDE Integration, Remote Targets

- OS: Linux (x86, Power, Tegra, Arm SBSA), Windows, macOS X (host only)
- GPUs: Volta+

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





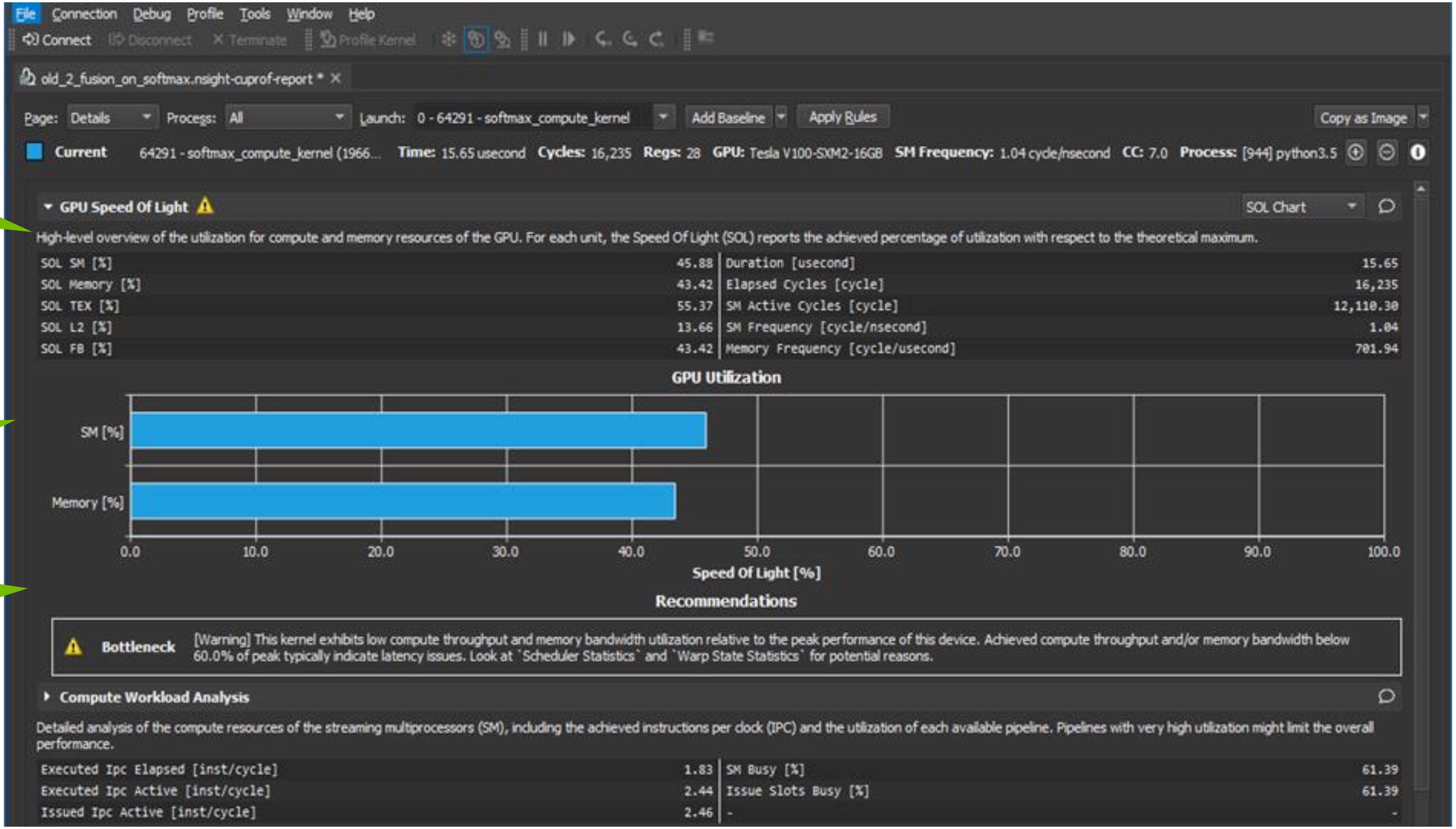


Nsight Compute GUI Interface

Targeted metric sections

Customizable data collection and presentation

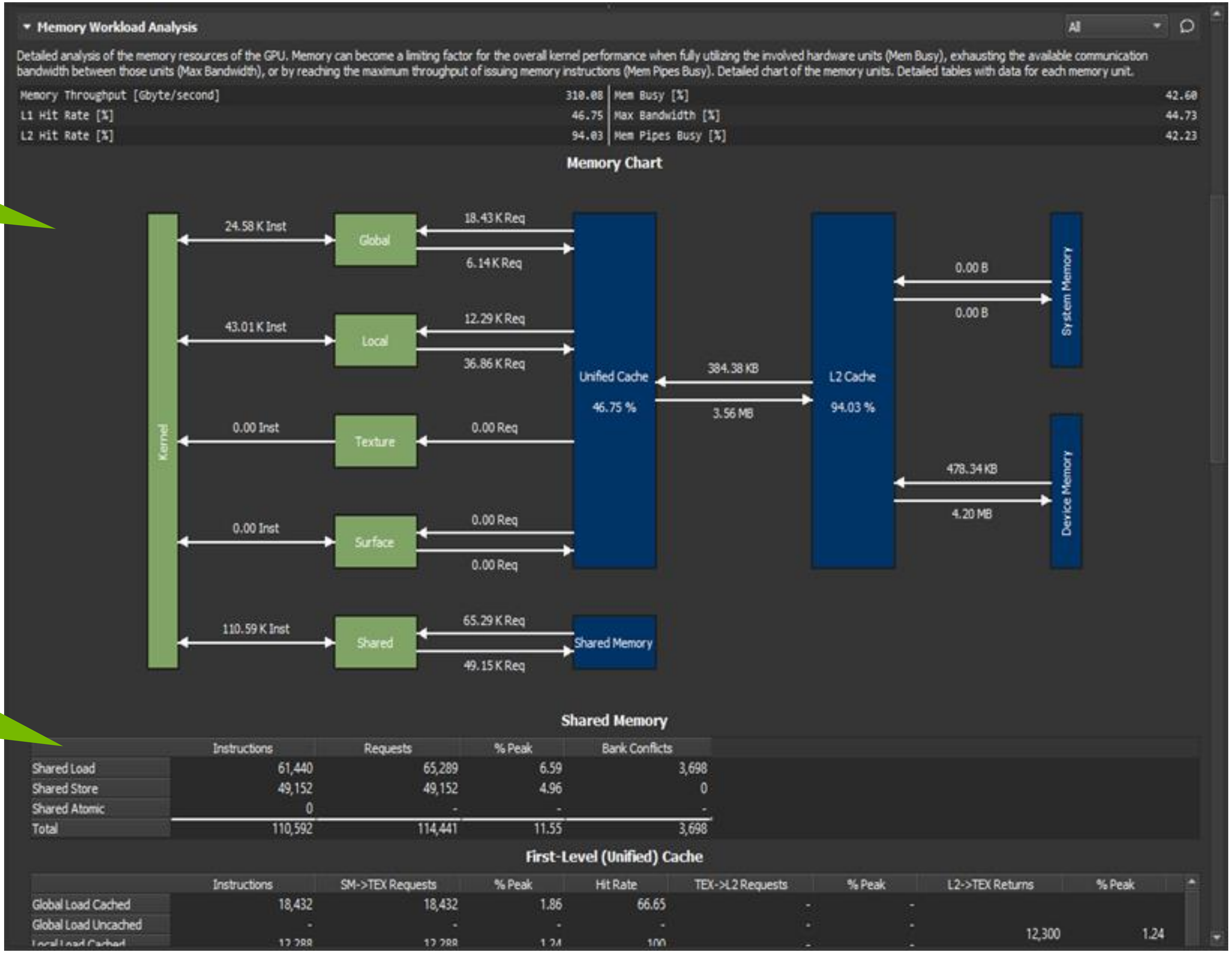
Built-in expertise for Guided Analysis and optimization

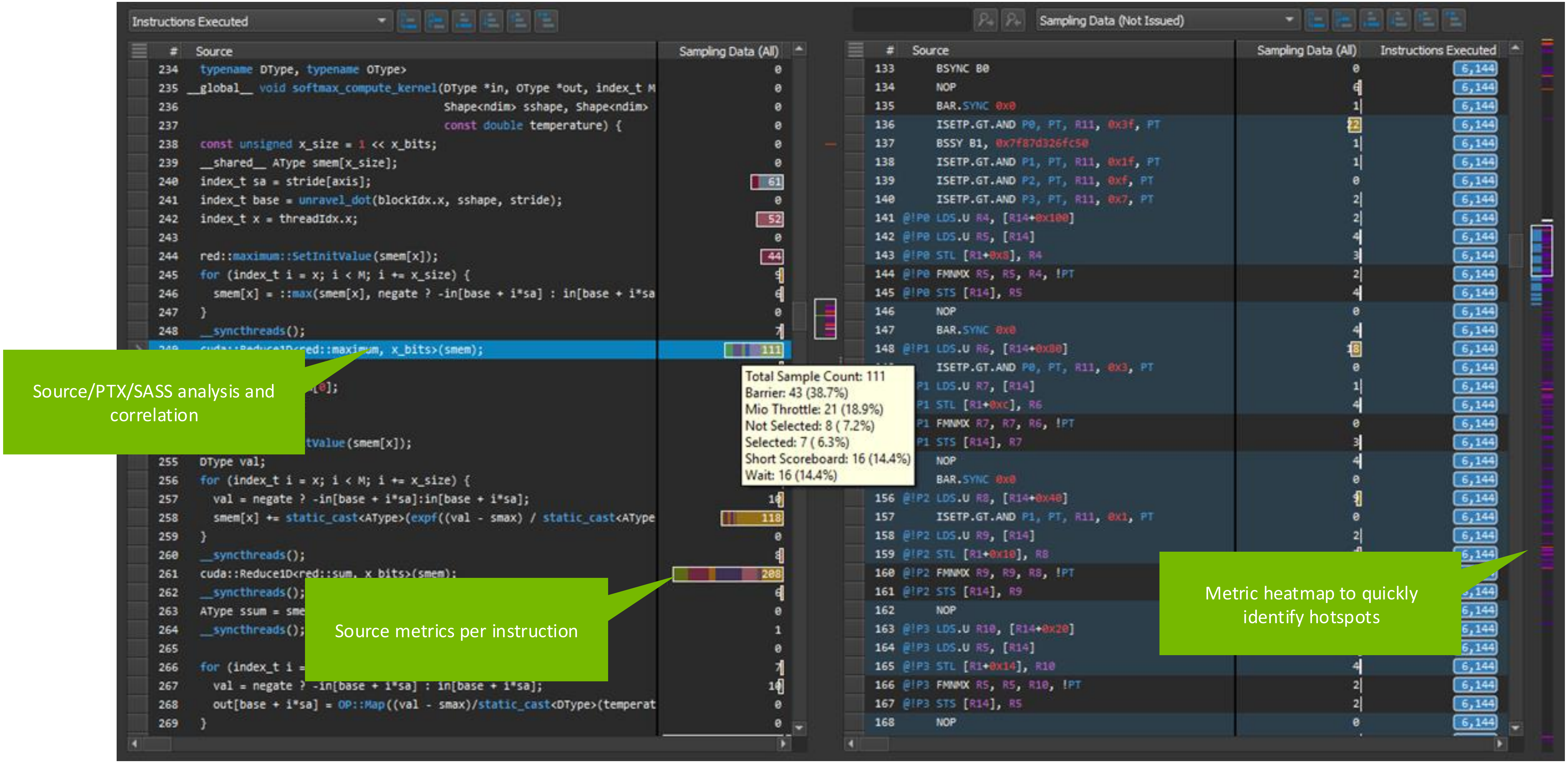


NVIDIA

Visual memory analysis chart

Metrics for peak performance ratios




Instructions Executed
Sampling Data (Not Issued)
Source
Sampling Data (All)
Instructions Executed
Source/PTX/SASS analysis and correlation
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 metrics per instruction
Metric heatmap to quickly identify hotspots

Kernel Profiles with Nsight Compute

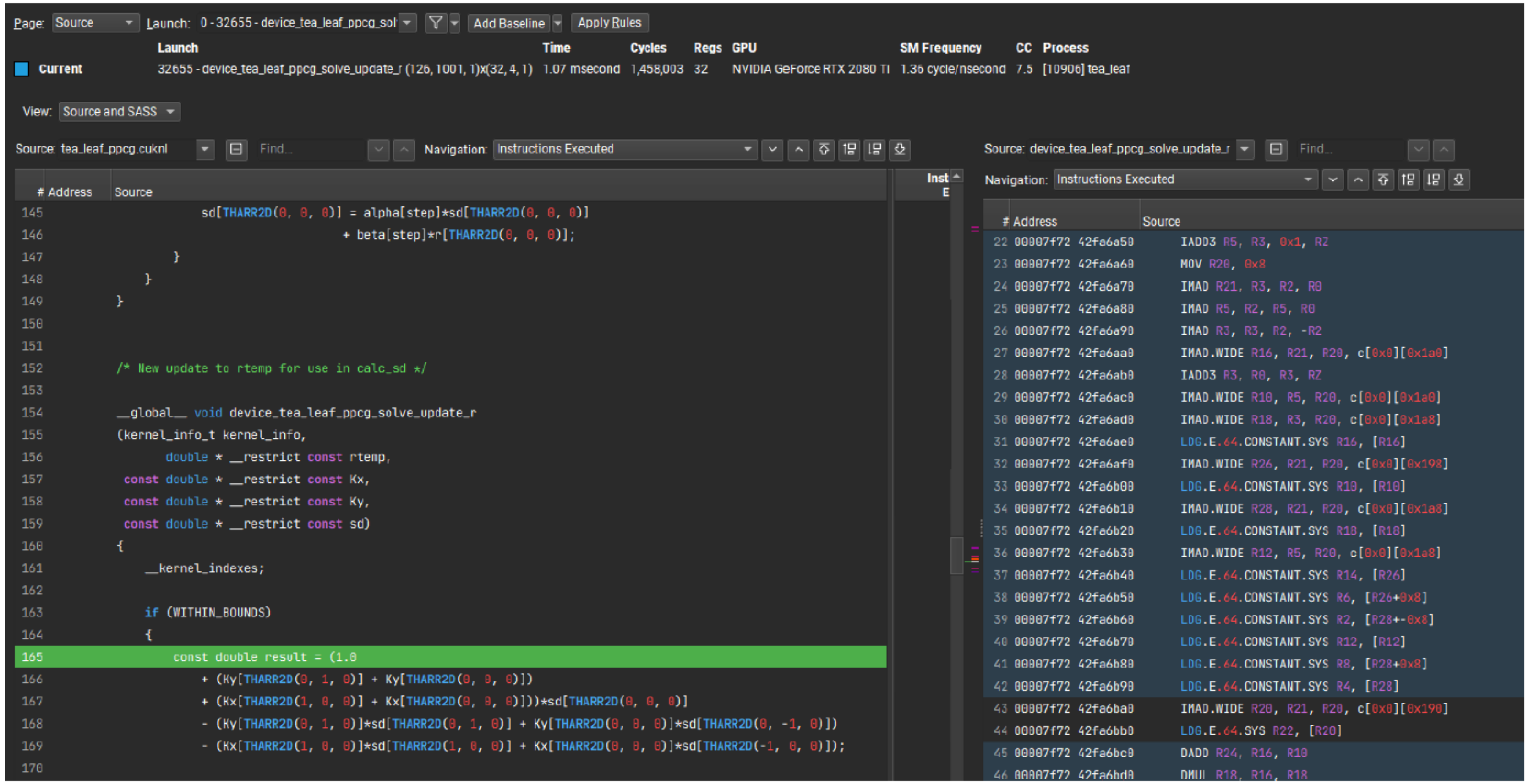
```
$ ncu –k mykernel –o report ./myapp.exe
```

- 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)

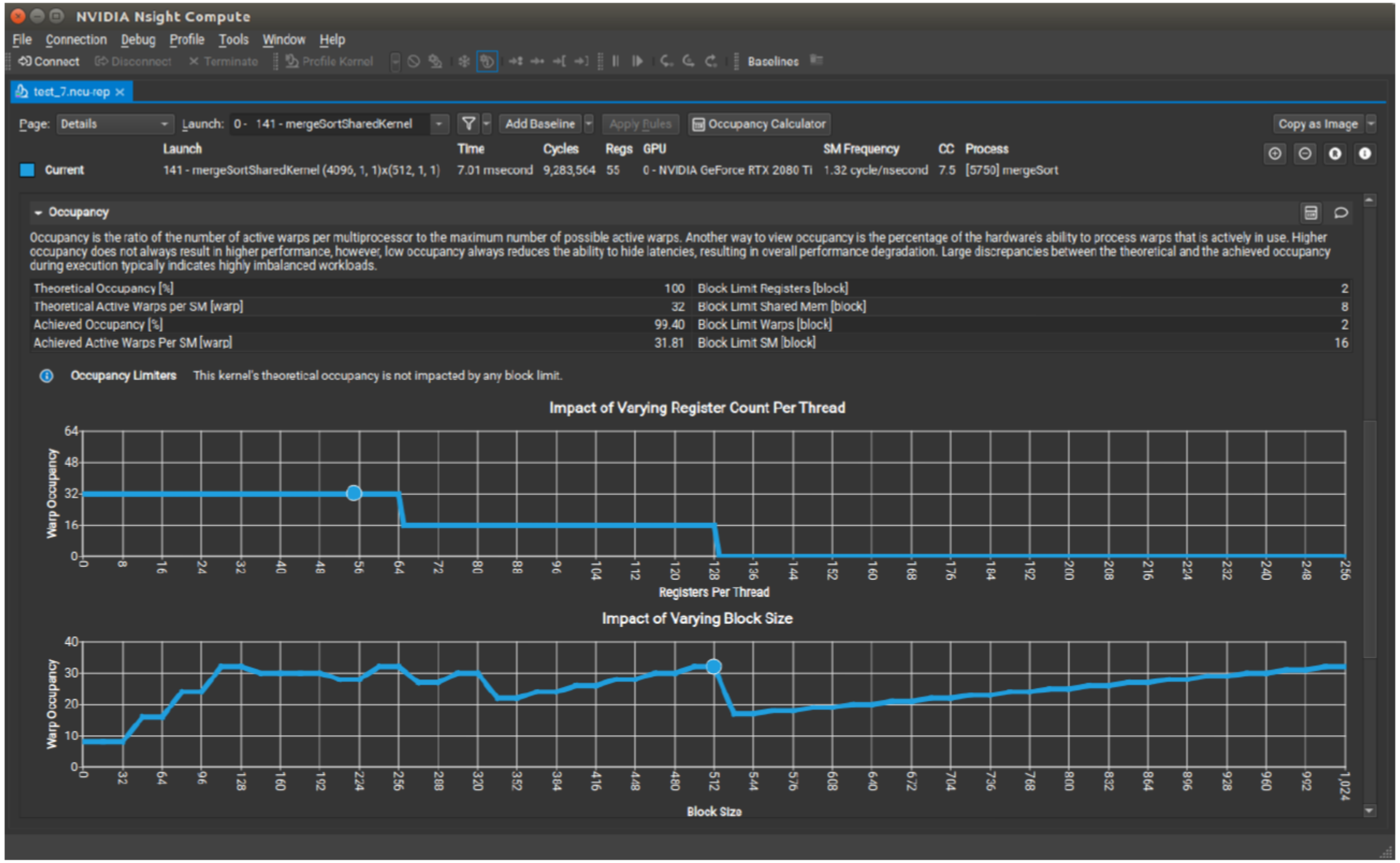
Standalone Source Viewer

- 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



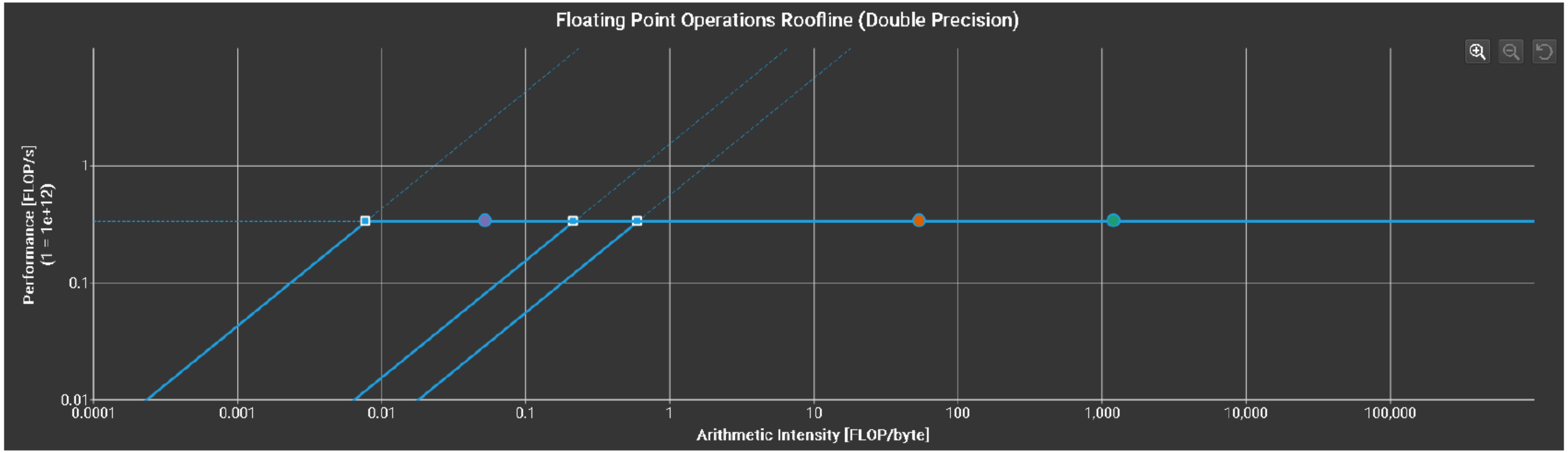
Occupancy Calculator

Model hardware usage and identify limiters

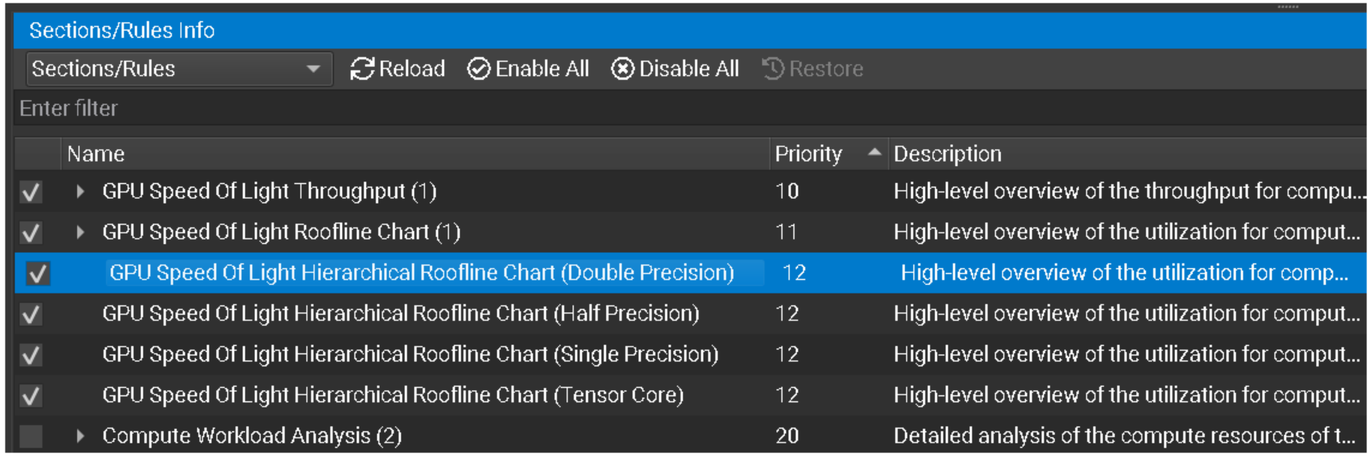


- Model theoretical hardware usage
- Understand limitations from hardware vs. kernel parameters
- Configure model to vary HW and kernel parameters
- Opened from an existing report or as a new activity

Hierarchical Roofline

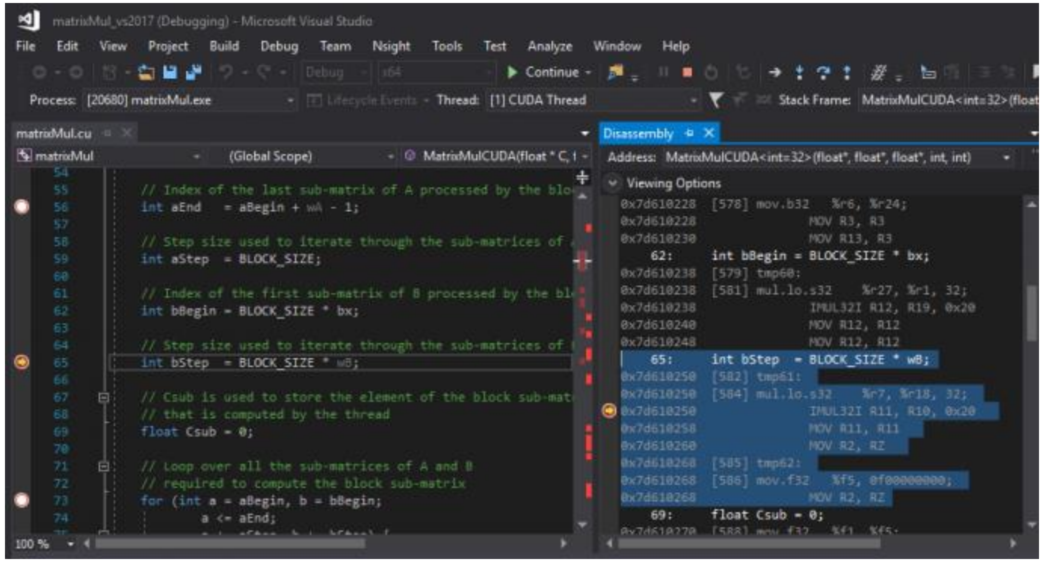


- Visualize multiple levels of the memory hierarchy
- Identify bottlenecks caused by memory limitations
- Determine how modifying algorithms may (or may not) impact performance

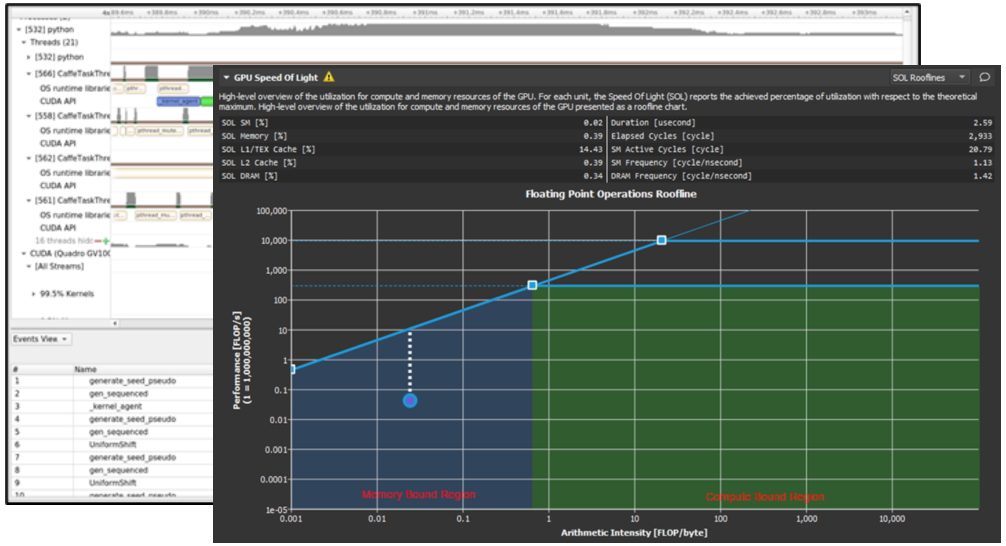


Developer Tools

Debuggers: cuda-gdb, Nsight Visual Studio Edition



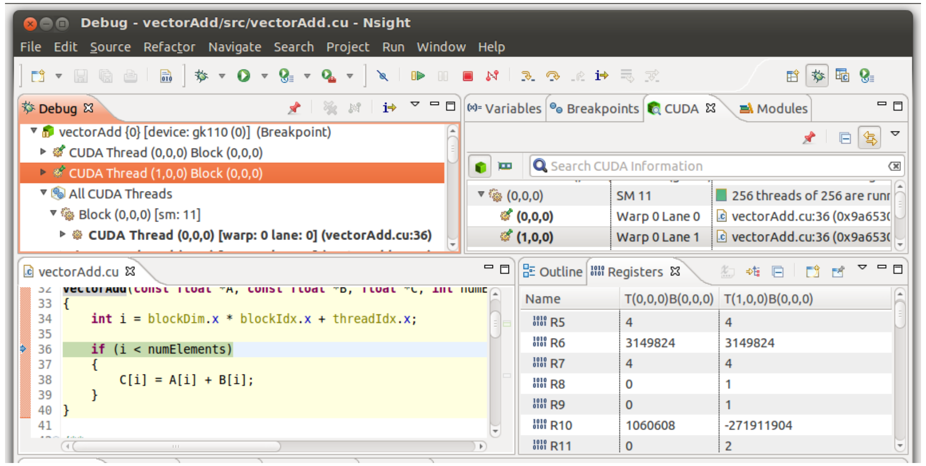
Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)



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
```

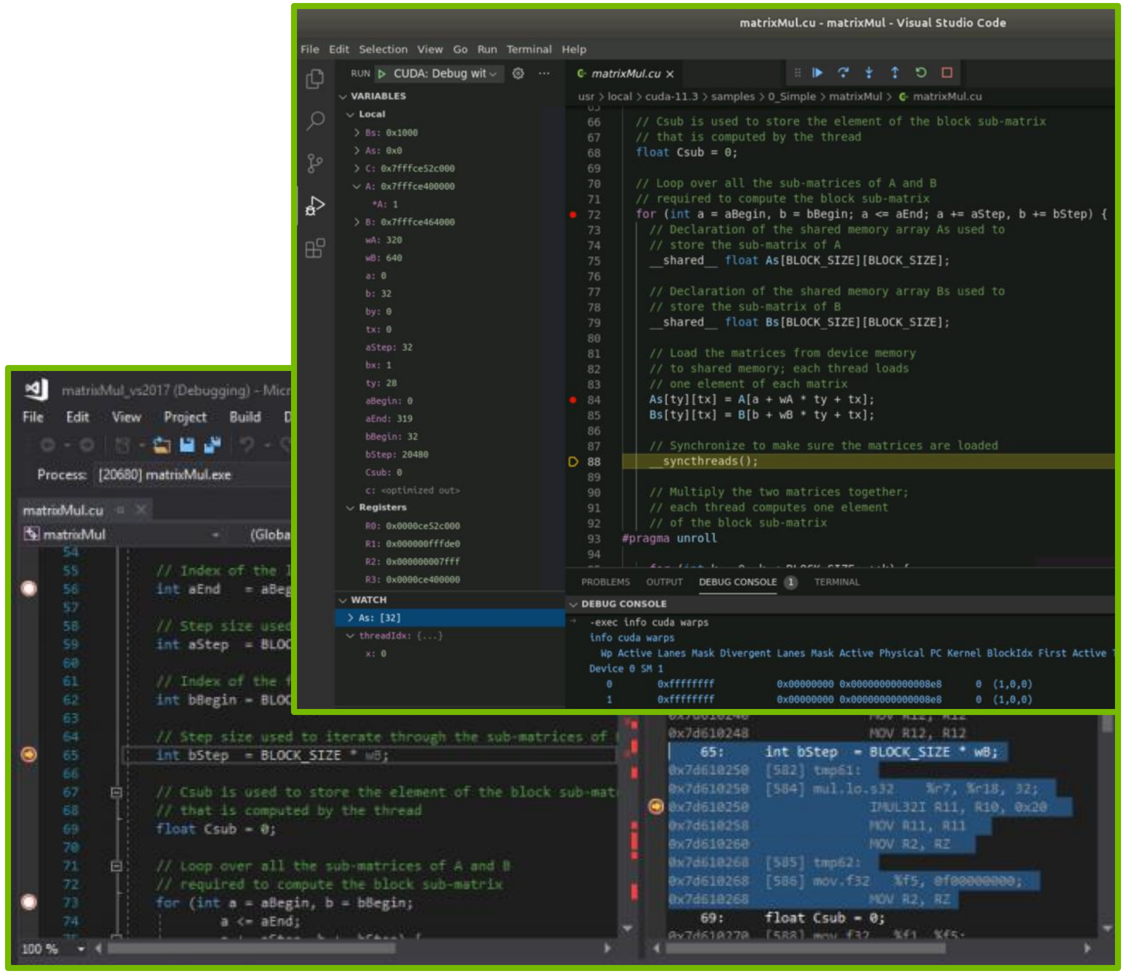
IDE integrations: Nsight Eclipse Edition, Nsight Visual Studio Edition, Nsight Visual Studio **Code** Edition



Compute Debuggers

Debug GPU kernels running on device

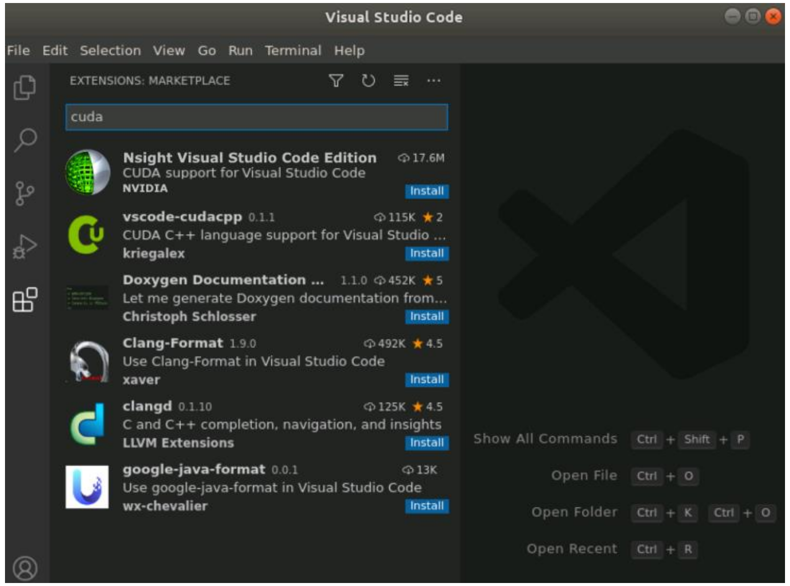
- CUDA GDB
  - CPU + GPU CUDA kernel debugger
  - Supports stepping, breakpoints, in-line functions, variable inspection etc…
  - Built on GDB and uses many of the same CLI commands
  - Local/Remote connection support
- Nsight Visual Studio Edition
  - IDE integration for Visual Studio
  - Build and Debug CPU+GPU code from Visual Studio
- Nsight Visual Studio Code Edition
  - New IDE integration for VS Code
  - Build and Debug CPU+GPU code from Visual Studio Code
  - Remotely target Linux targets from Windows or Linux
- Nsight Eclipse Edition
  - IDE integration for Eclipse
  - Build and Debug CPU+GPU code from Eclipse

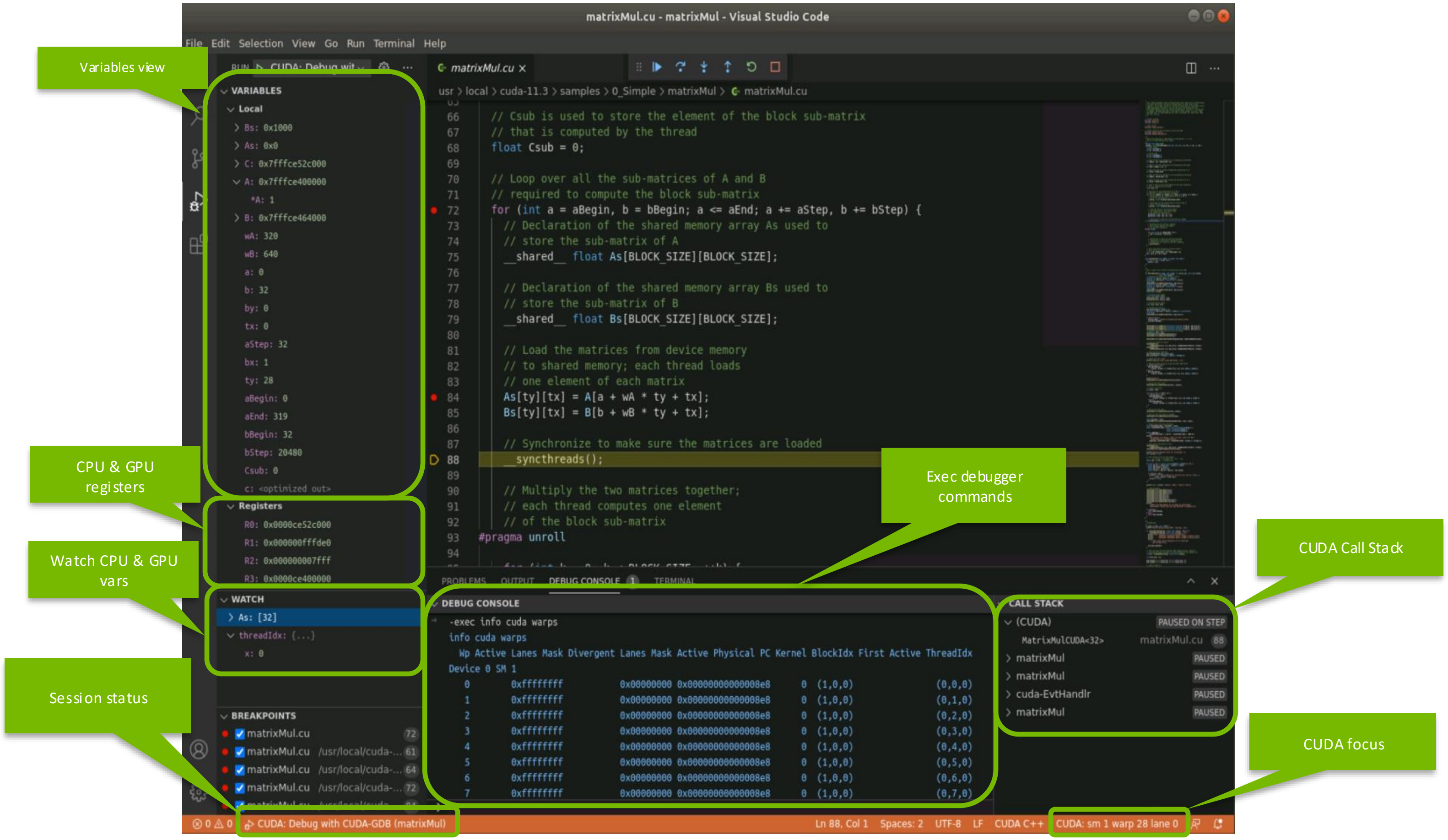


Nsight Visual Studio Code Edition

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

Compute Sanitizer

Automatically Scan for Bugs and Memory Issues

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

https://github.com/NVIDIA/compute-sanitizer-samples

```
$ make run_memcheck
/usr/local/cuda/compute-sanitizer/compute-sanitizer --destroy-on-device-error kernel memcheck_demo
========= COMPUTE-SANITIZER
Mallocing memory
========= Invalid __global__ write of size 4 bytes
=========     at 0x70 in unaligned_kernel()
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x7f671ac00001 is misaligned
=========     and is inside the nearest allocation at 0x7fb654c00000 of size 4 bytes
=========     Saved host backtrace up to driver entry point at kernel launch time
=========     Host Frame: [0x2774ec]
=========                in /lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:__cudart803 [0xfccb]
=========                in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=========     Host Frame:cudaLaunchKernel [0x6a578]
=========                in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=========     Host Frame:cudaError cudaLaunchKernel<char>(char const*, dim3, dim3, void**, unsigned
=========                in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=========     Host Frame:__device_stub__Z16unaligned_kernelv() [0xb22e]
=========                in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=========     Host Frame:unaligned_kernel() [0xb28c]
=========                in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=========     Host Frame:run_unaligned() [0xaf55]
=========                in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=========     Host Frame:main [0xb0e2]
=========                in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=========     Host Frame:../sysdeps/nptl/libc_start_call_main.h:58:__libc_start_call_main [0x2dfd0]
=========                in /lib/x86_64-linux-gnu/libc.so.6
```

Compute Sanitizer

Reading a Memcheck Example Report

Address space

Type of access

Access size

Access location

Faulty thread

Faulty address and nearest allocation

Device and host backtraces

```
========= Invalid __global__ write of size 4 bytes
=========     at 0xb0                ub/compute-sanitizer-samples/Memcheck/memcheck_demo.cu:39:out_of_bounds_function()
=========     by threa
=========     Address 0x87654320 is out of bounds
=========                          ytes before the nearest allocation at 0x7f953da00000 of size 1,024 bytes
=========     Device Frame:/home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo.cu:44:out_of_bounds_kernel() [0x30]
=========     Saved host backtrace up to driver entry point at kernel launch time
=========     Host Frame: [0x2774ec]
=========                in /lib/x86_64-linux-gnu/libcuda.so.1
=========     Host Frame:__cudart803 [0xfccb]
=========                in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=========     Host Frame:cudaLaunchKernel [0x6a578]
=========                in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
```

Other Resources

Continue your learning journey. Keep engaged after the event. www.openhackathons.org

[Open Hackathons GitHub]()

Explore additional Bootcamp materials and resources

[TACC Open Hackathon]()

Deadline: August 7, 2024

- Advance and accelerate your science
- Work with dedicated mentors
- Access the latest systems

[NVIDIA DLI]()

Resources for diverse training

Explore AI, accelerated computing, data science, graphics, and more.

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

NVIDIA