NVIDIA PROFILING TOOLS

KRISTOPHER KEIPERT

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

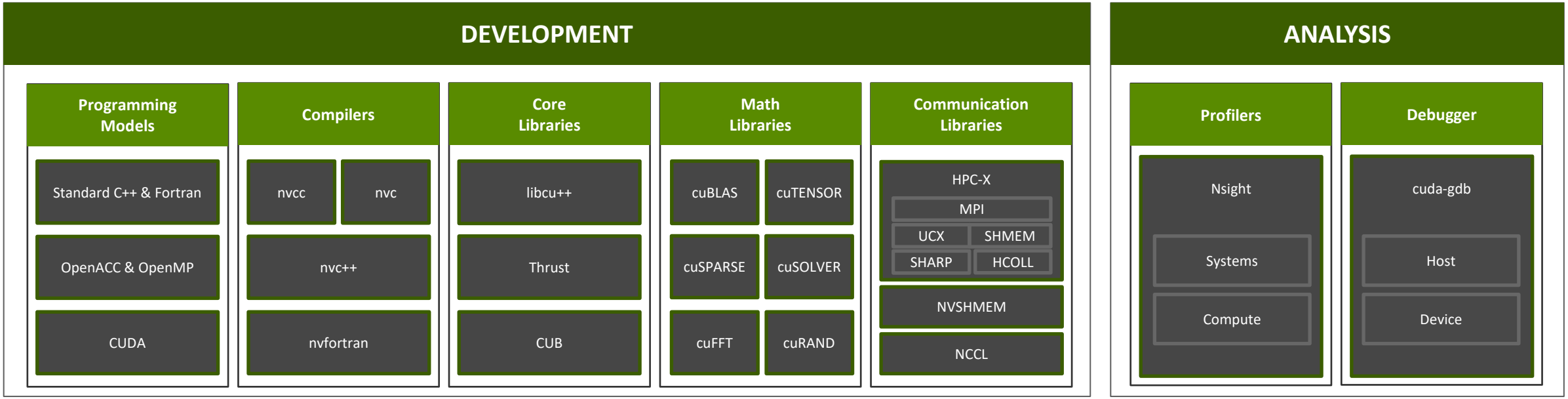
```
__global__
void saxpy(int n, float a,
           float *x, float *y) {
  int i = blockIdx.x*blockDim.x +
          threadIdx.x;
  if (i < n) y[i] += a*x[i];
}

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



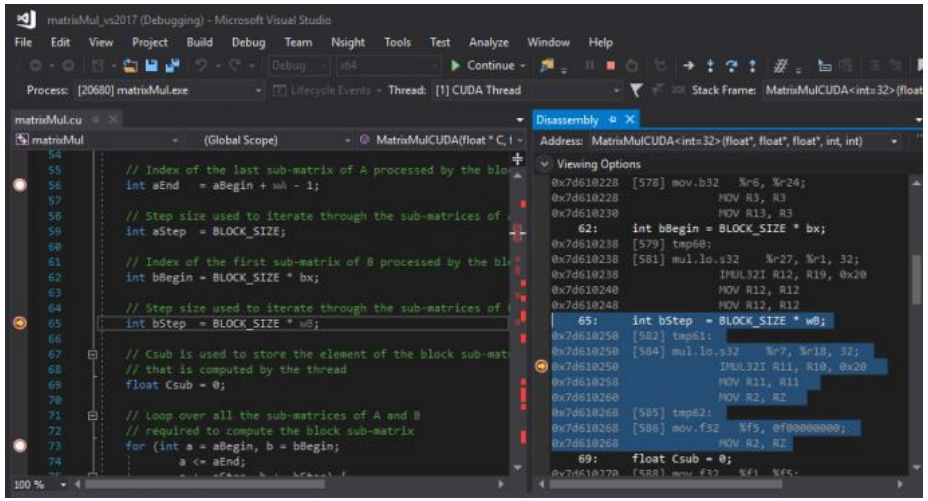
Develop for the NVIDIA Platform: GPU, CPU and Interconnect

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

7-8 Releases Per Year | Freely Available

DEVELOPER TOOLS

Debuggers: cuda-gdb, Nsight Visual Studio Edition

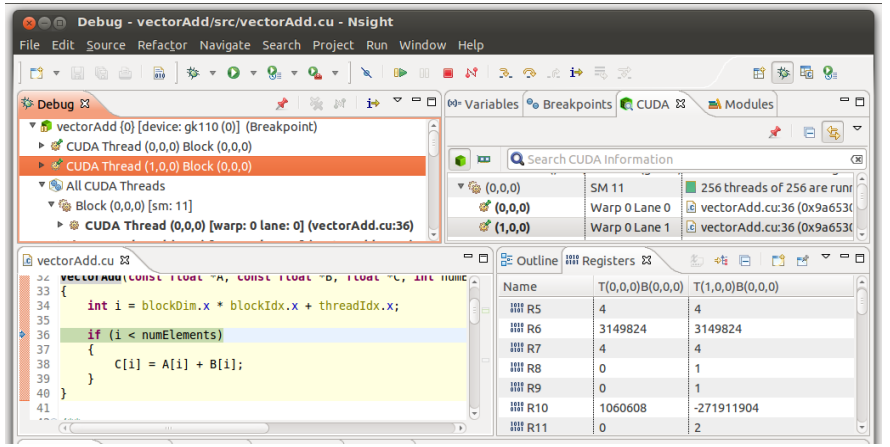


Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)

Correctness Checker:: Compute Sanitizer

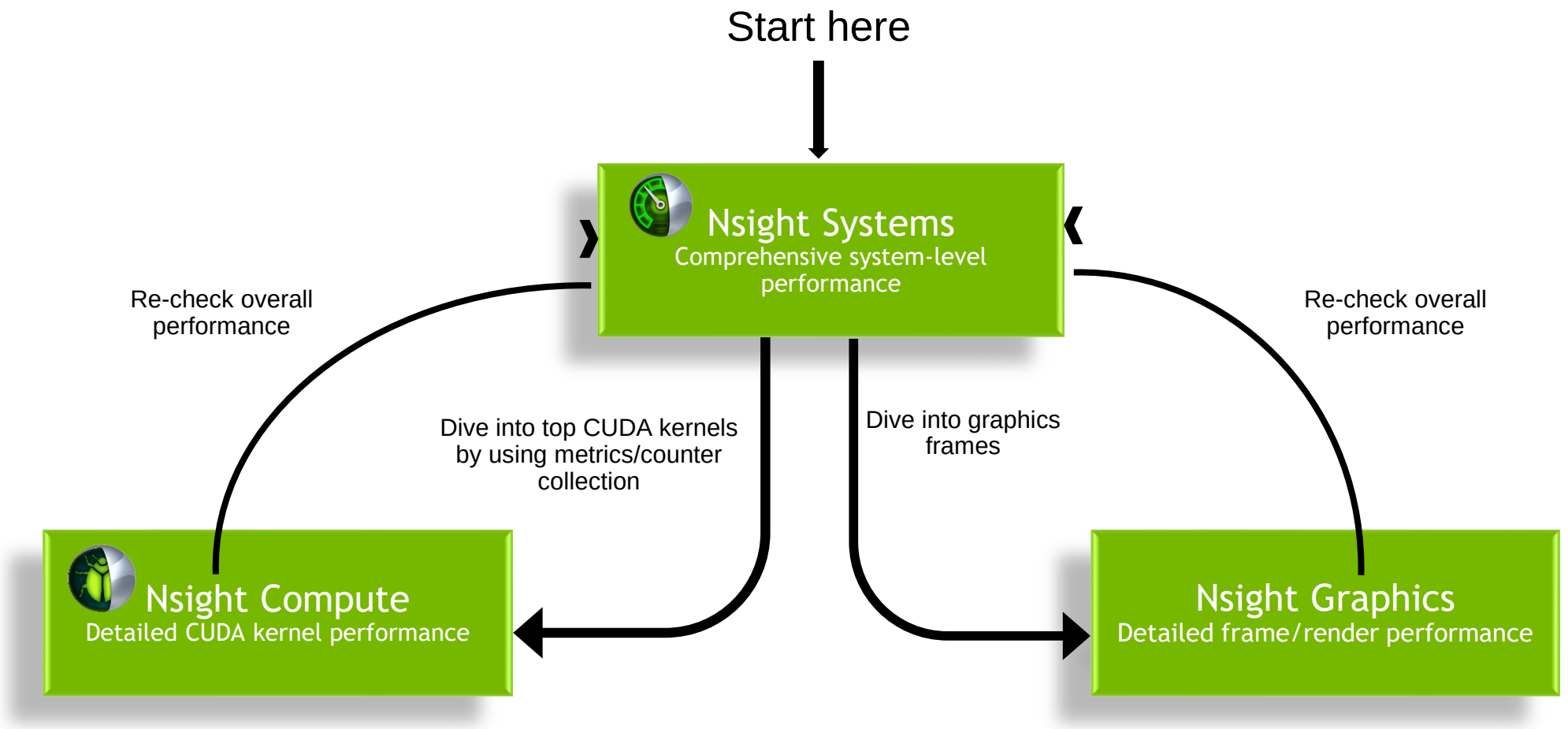
```
$ compute-sanitizer --leak-check full memcheck_demo
========= COMPUTE-SANITIZER
Mallocing memory
Running unaligned_kernel
Ran unaligned_kernel: no error
Sync: no error
Running out_of_bounds_kernel
Ran out_of_bounds_kernel: no error
Sync: no error
========= Invalid __global__ write of size 4 bytes
=========     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

NSIGHT TOOLS WORKFLOW



NSIGHT SYSTEMS

System Profiler

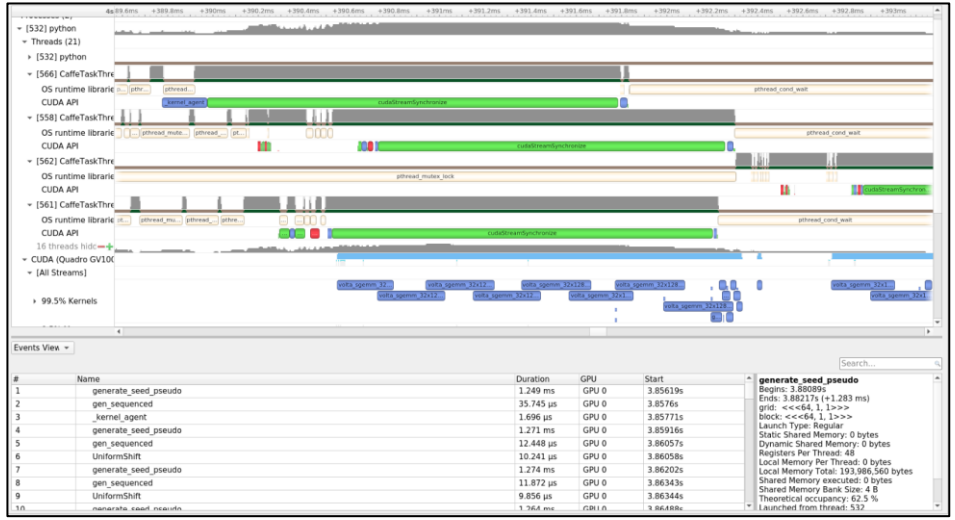
Key Features:

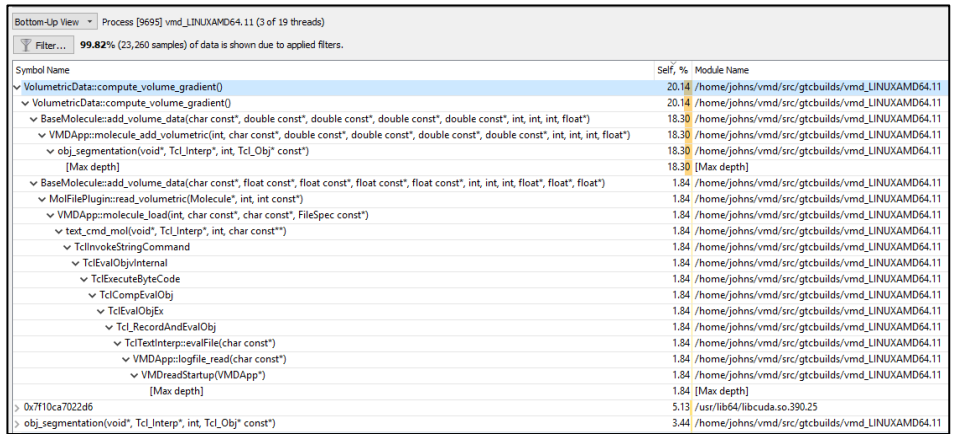
- System-wide application algorithm tuning
  - Multi-process tree support
- Locate optimization opportunities
  - Visualize millions of events on a very fast GUI timeline
  - Or gaps of unused CPU and GPU time
- Balance your workload across multiple CPUs and GPUs
  - CPU algorithms, utilization and thread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

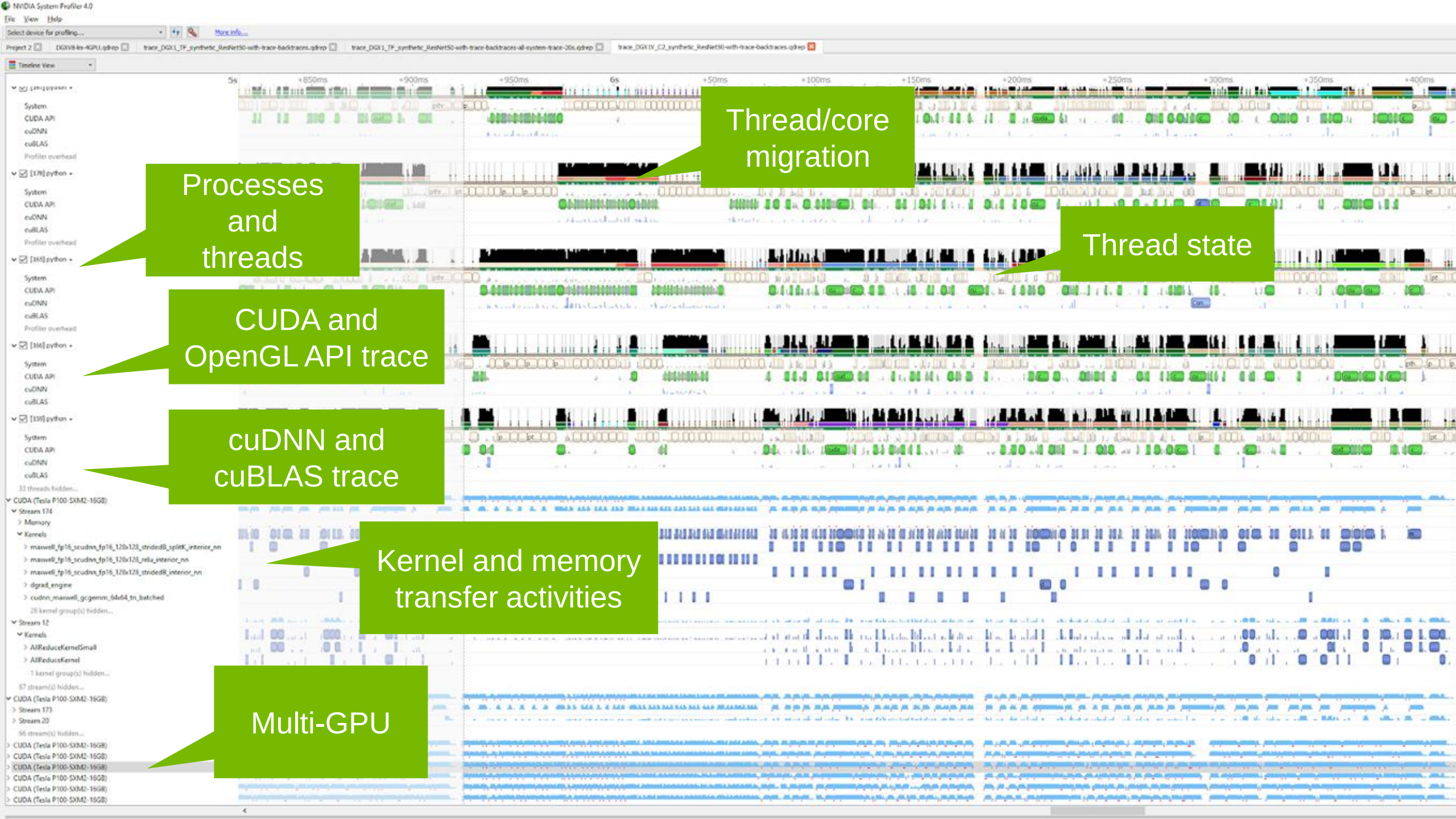
OS: Linux (x86, Power, Arm SBSA, Tegra), Windows, MacOSX (host)

GPUs: Pascal+

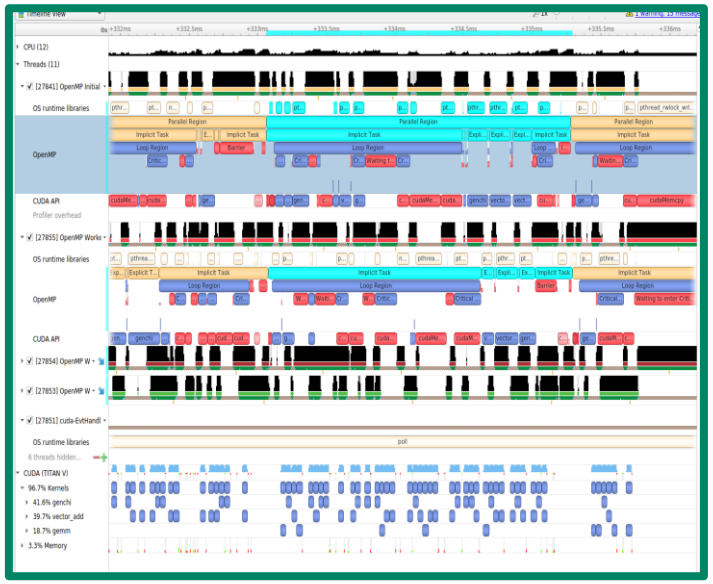
Docs/product: https://developer.nvidia.com/nsight-systems

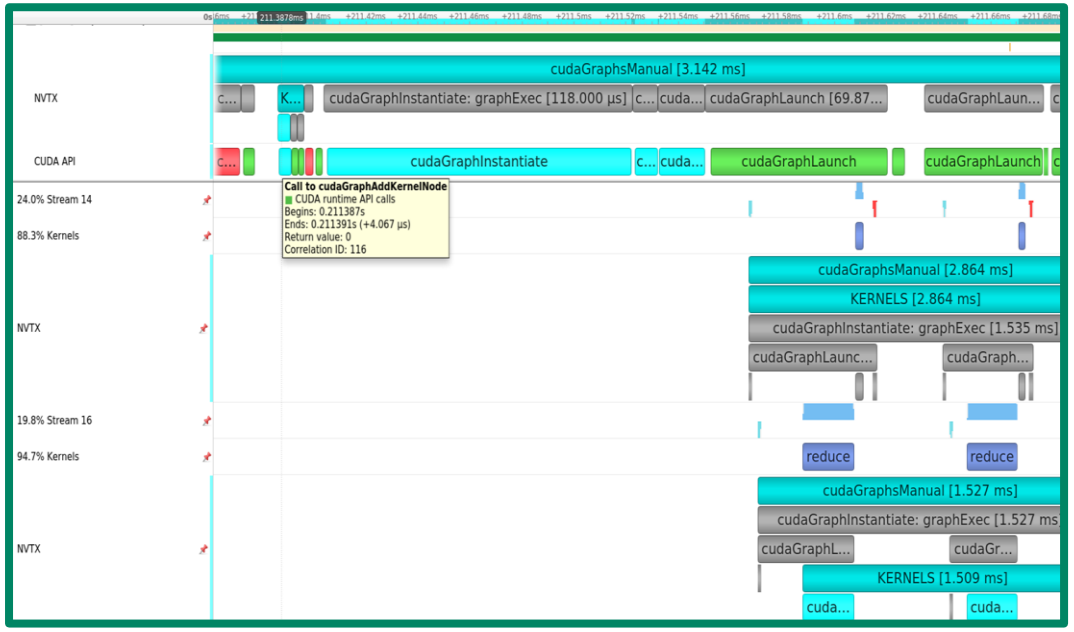




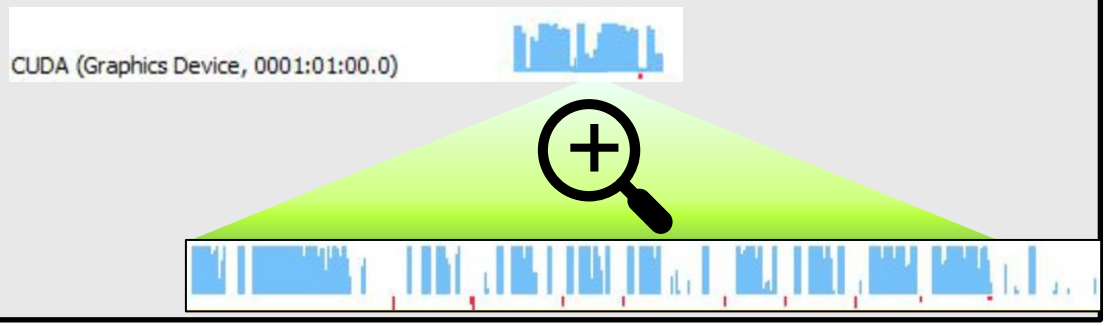

Thread/core migration
Processes and threads
Thread state
CUDA and OpenGL API trace
cuDNN and cuBLAS trace
Kernel and memory transfer activities
Multi-GPU

ZOOM/FILTER TO EXACT AREAS OF INTEREST



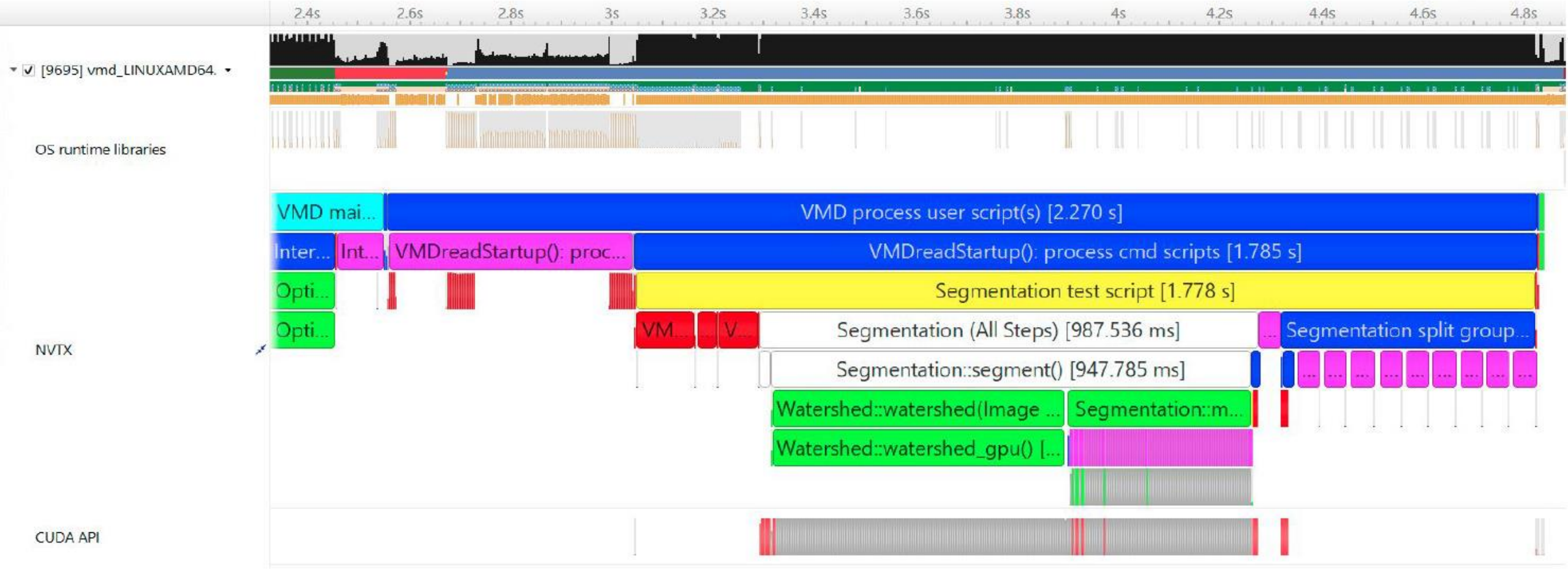


- Zoom in valleys to find gaps!



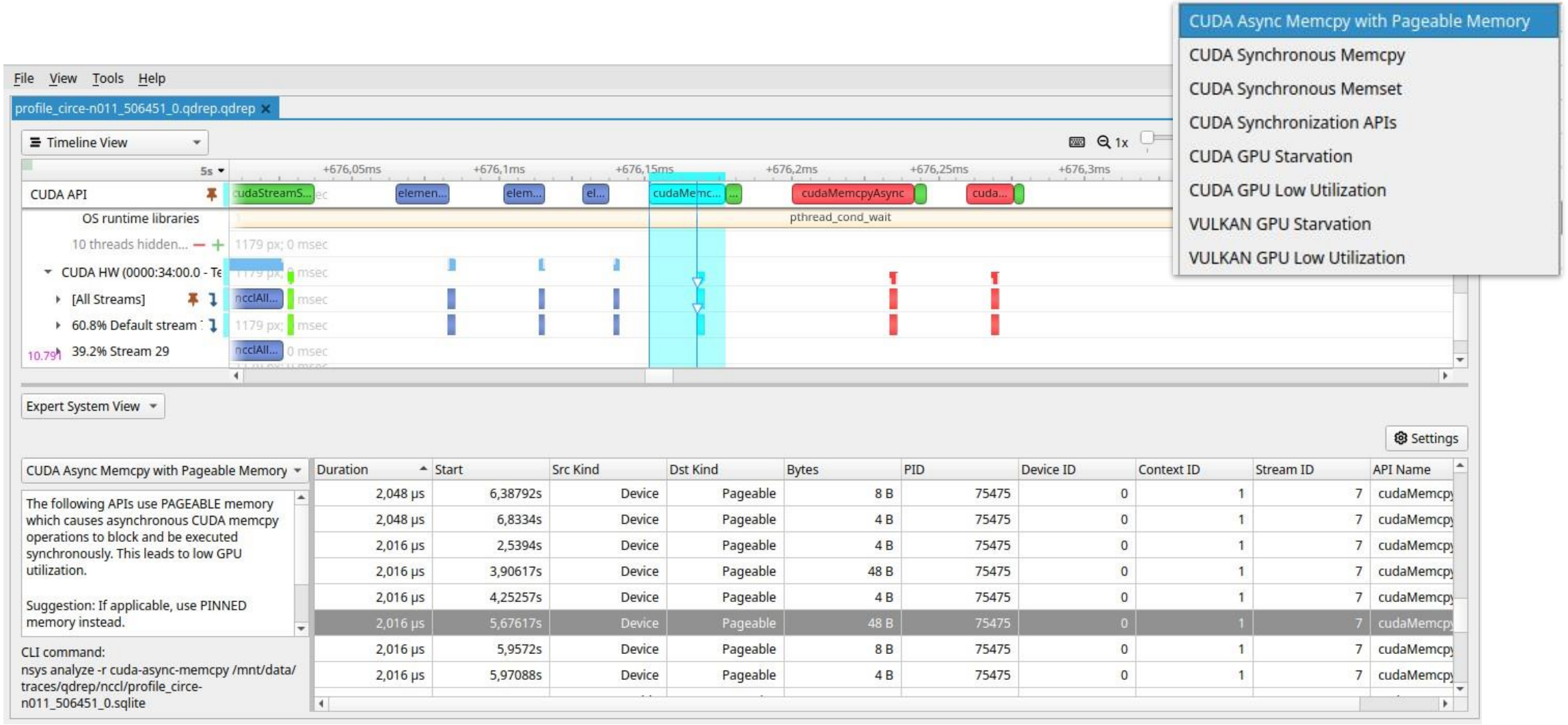
NVTX: NVIDIA TOOLS EXTENSIONS

Code Annotation API



EXPERT SYSTEMS & STATISTICS

Built-in Data Analytics with Advice

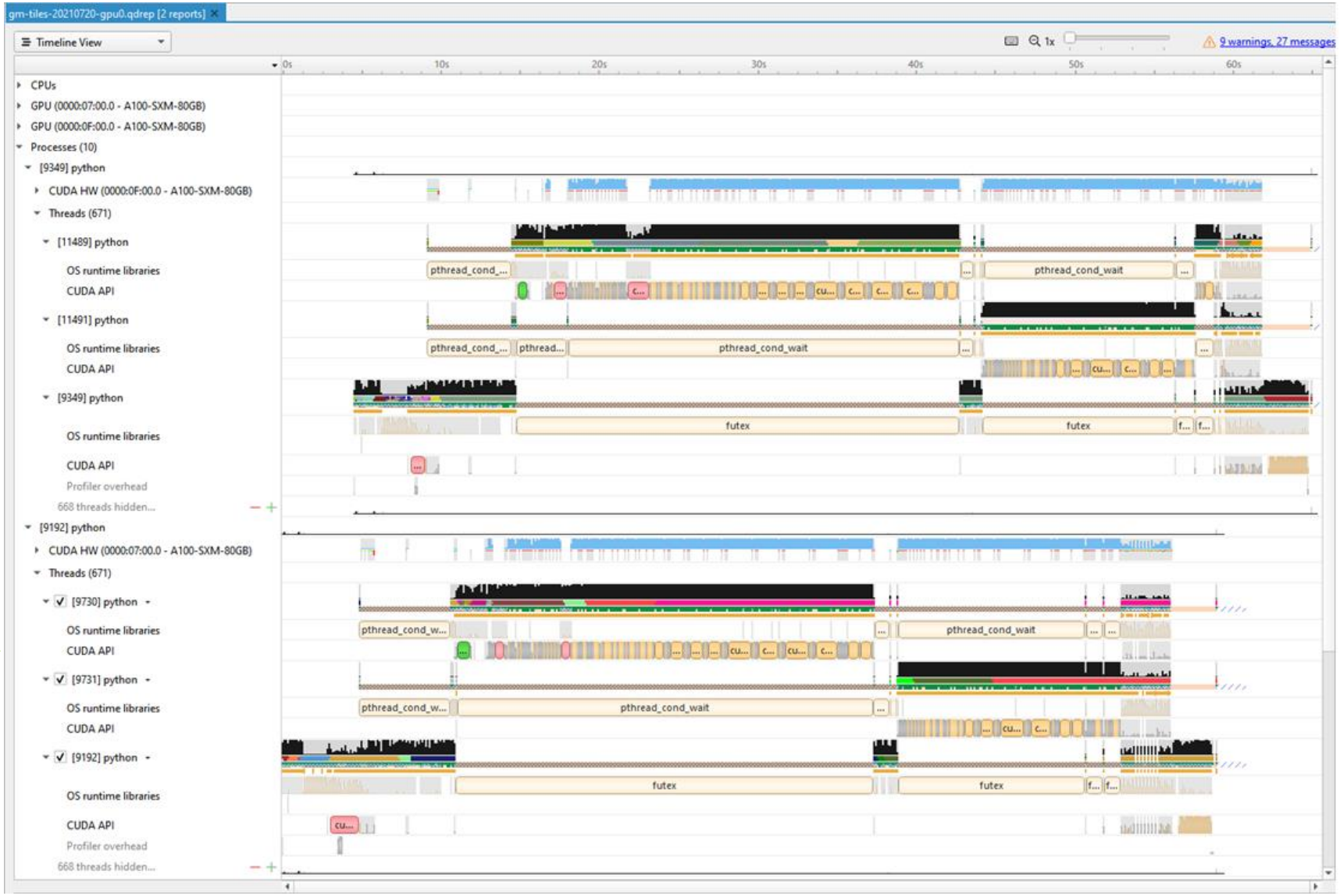


MULTI-REPORT TILING

Visualize More Parallel Activity

Open multiple reports

Loaded on same timeline based on wall-clock



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

NSIGHT COMPUTE

Kernel Profiling Tool

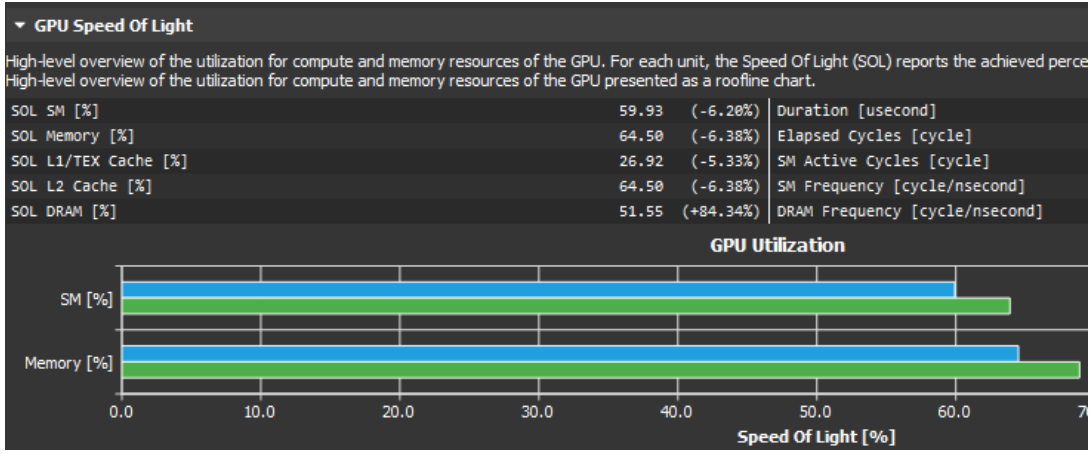
Key Features:

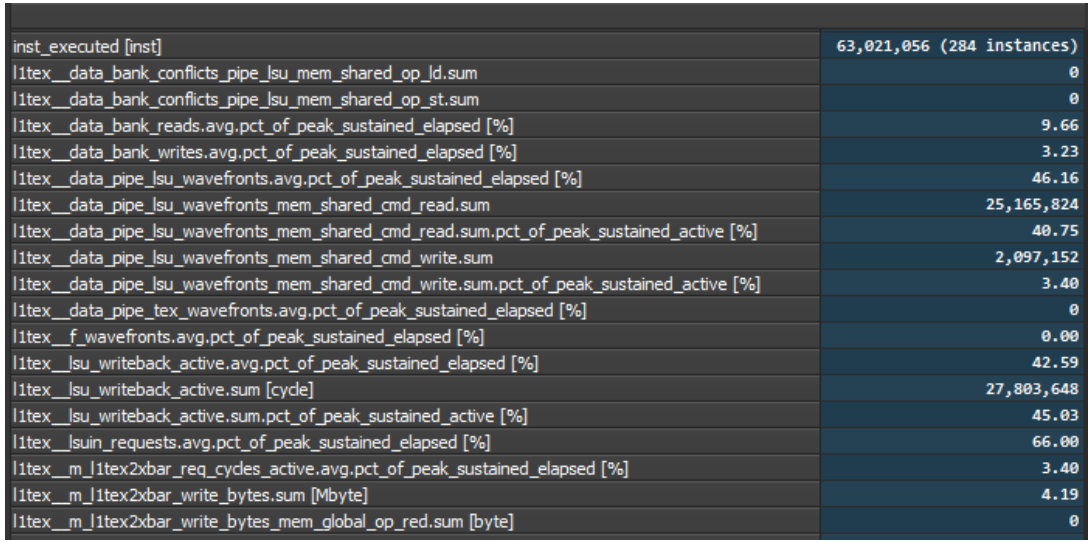
- Interactive CUDA API debugging and kernel profiling
- Built-in rules expertise
- Fully customizable data collection and display
- Command Line, Standalone, IDE Integration, Remote Targets

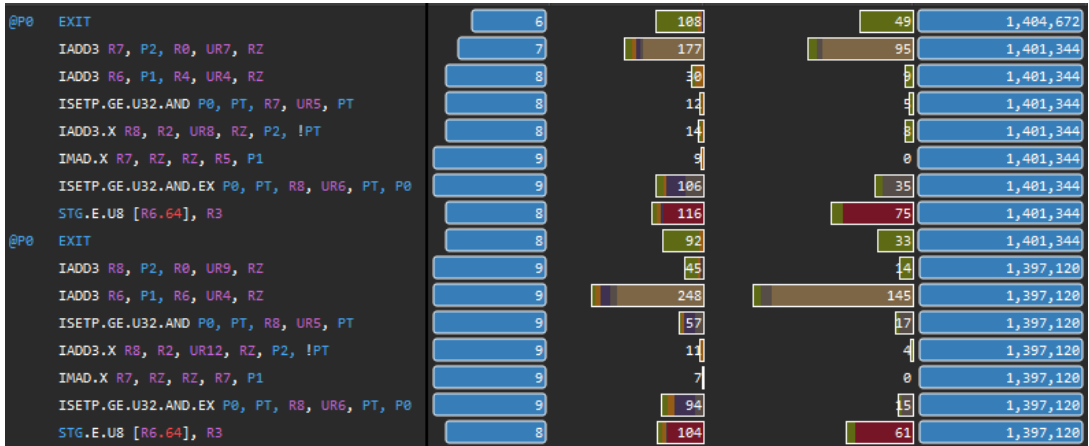
OS: Linux (x86, Power, Tegra, Arm SBSA), Windows, MacOSX (host only)

GPUs: Volta+

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



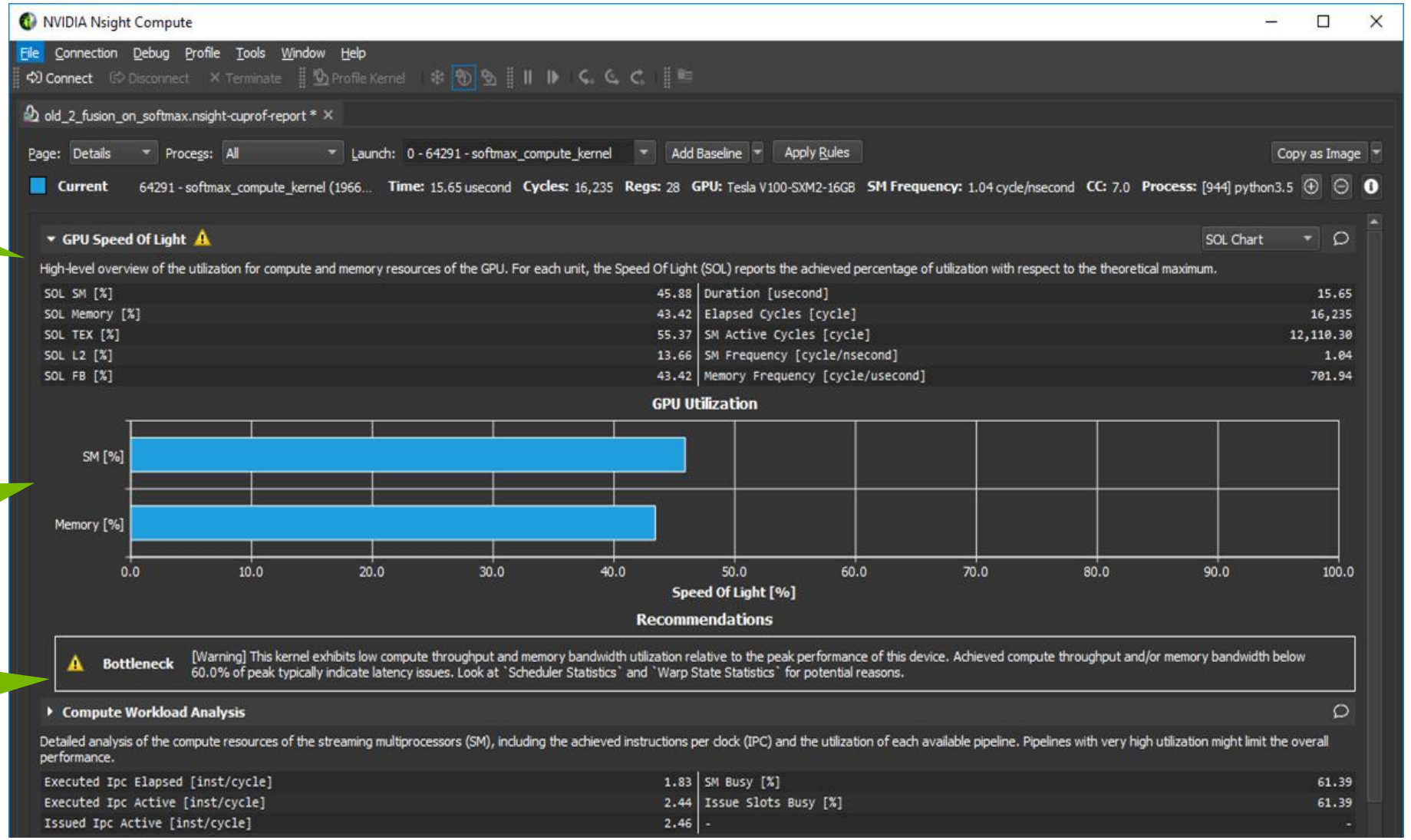




Targeted metric sections

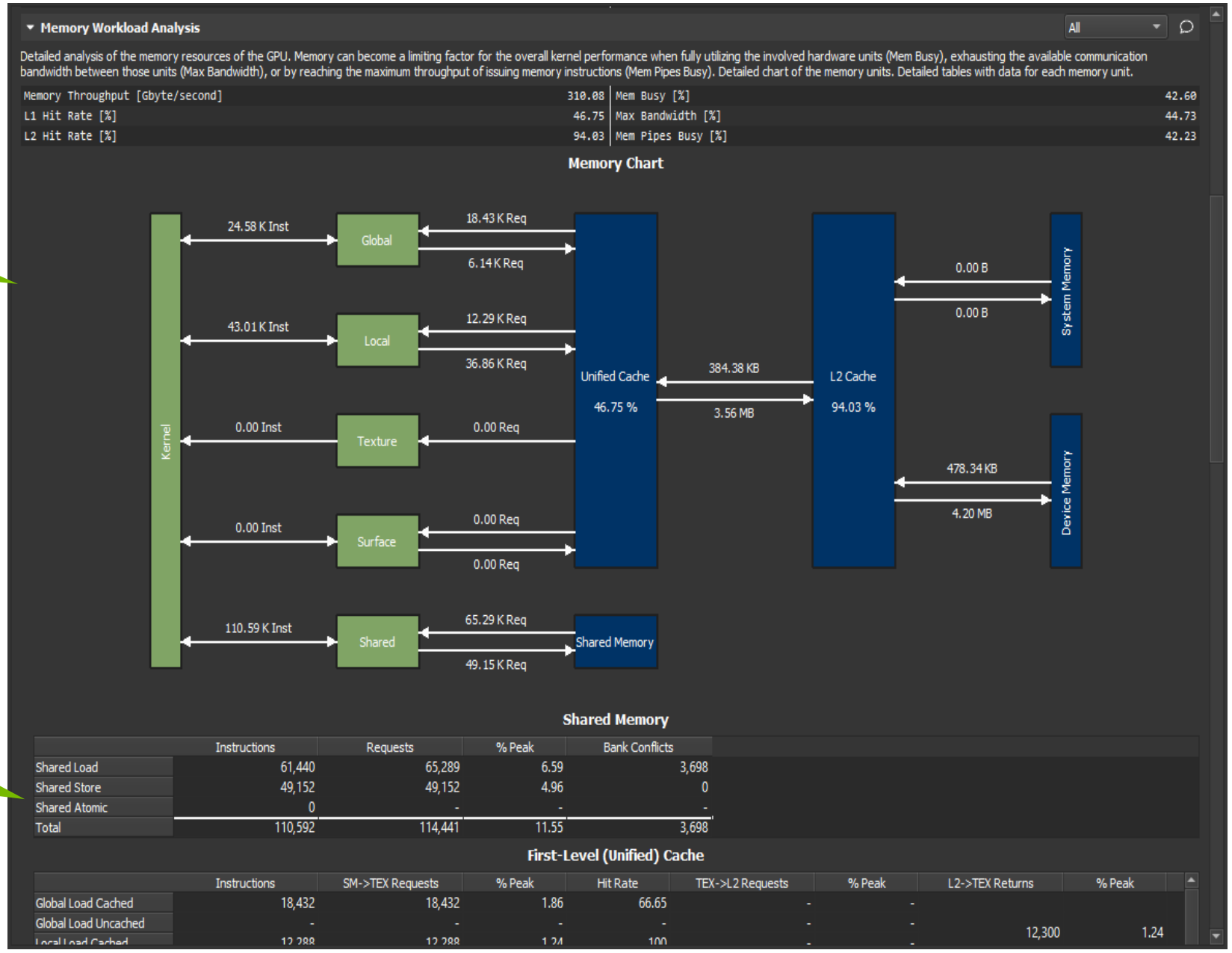
Customizable data collection and presentation

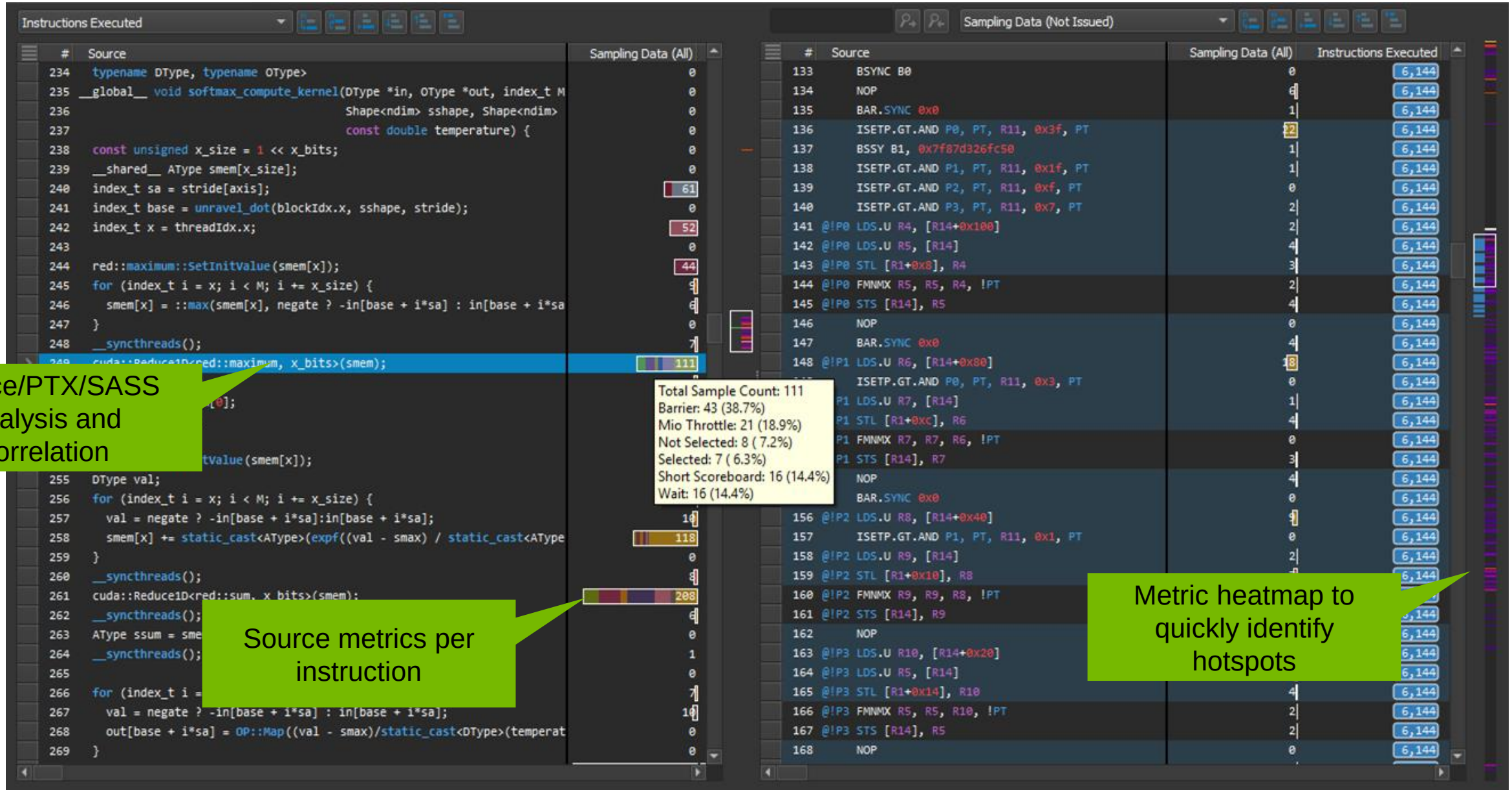
Built-in expertise for Guided Analysis and optimization



Visual memory analysis chart

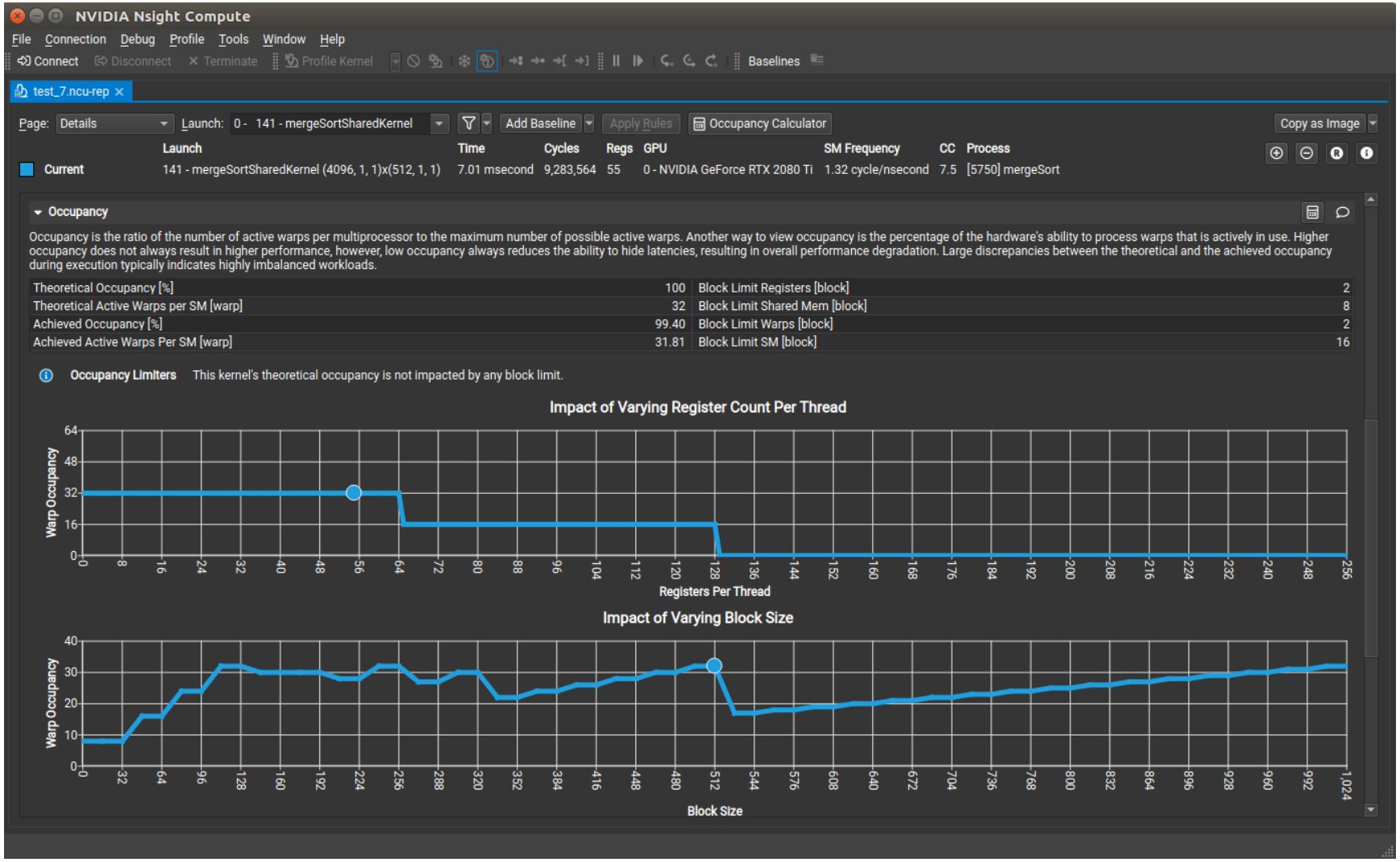
Metrics for peak performance ratios




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

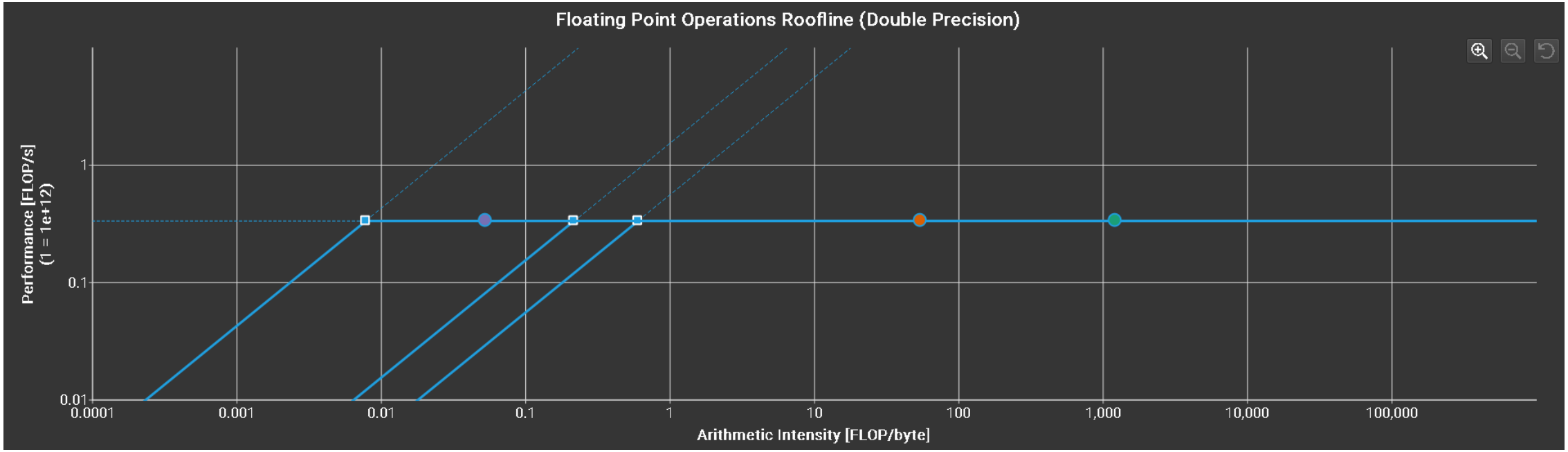
OCCUPANCY CALCULATOR

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...

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)

CUDA-GDB

Command-Line and IDE Back-End Debugger

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

```
(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

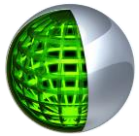
Automatically Scan for Bugs and Memory Issues

- Compute Sanitizer checks correctness issues via sub-tools:
  - **Memcheck** – Memory access error and leak detection 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

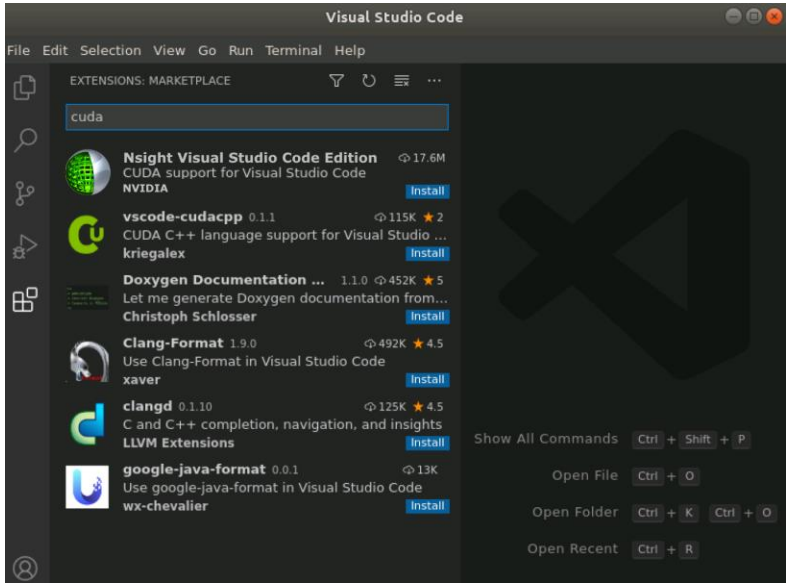
```
$ make run_memcheck
/usr/local/cuda/compute-sanitizer/compute-sanitizer --destroy-on-device-error kernel memcheck_demo
========= COMPUTE-SANITIZER
Mallocing memory
========= Invalid __global__ write of size 4 bytes
=========     at 0x70 in unaligned_kernel()
=========     by thread (0,0,0) in block (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
```

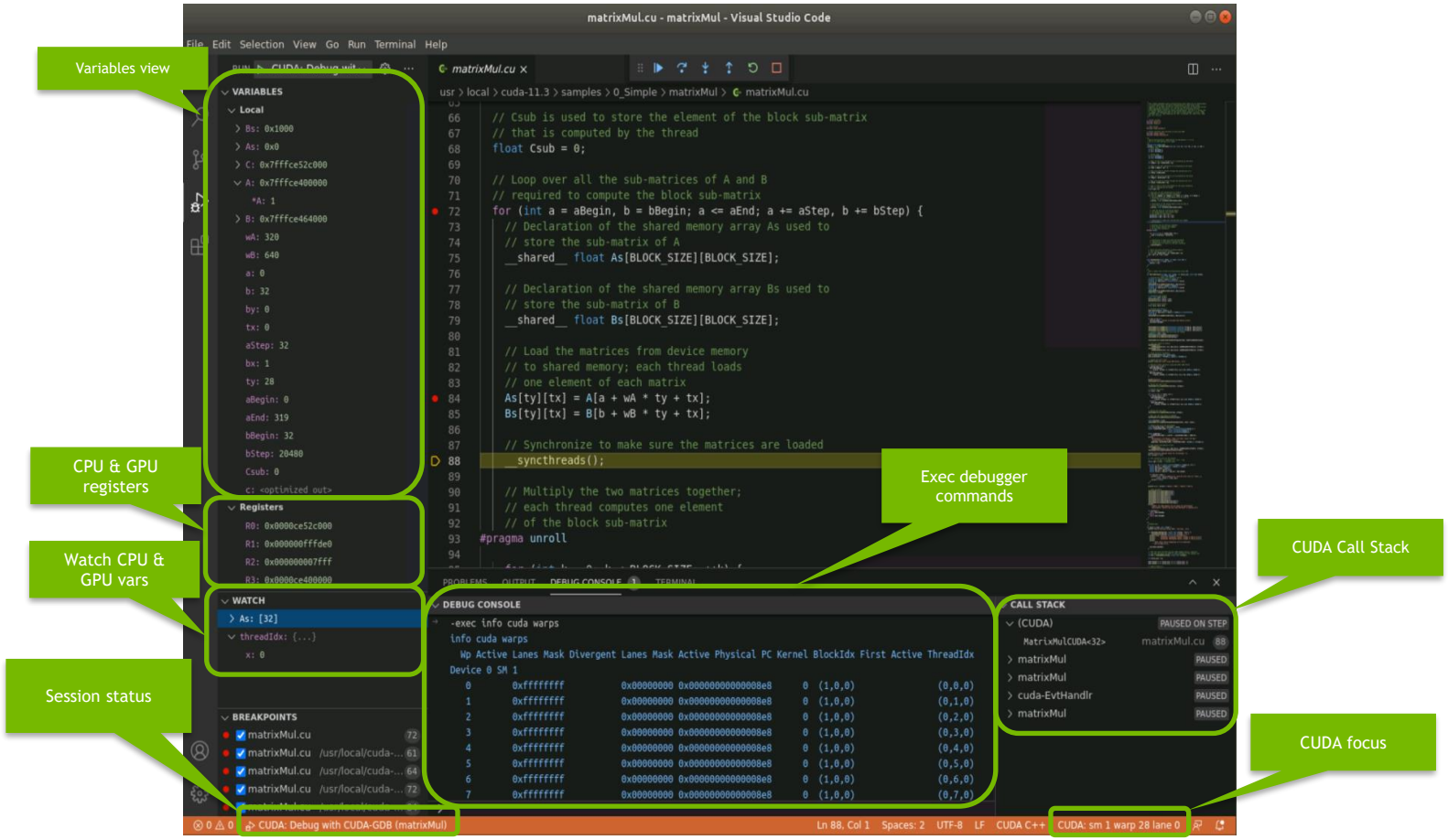
NSIGHT VISUAL STUDIO CODE EDITION



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

ADDITIONAL RESOURCES

- Sessions
  - A41100 - CUDA: New Features and Beyond
  - A41131 - Developing Efficient CUDA Kernels for Fourth-Generation Tensor Cores
- Labs
  - DLIT41277 - Optimizing CUDA Machine Learning Codes with Nsight Profiling Tools
  - DLIT41274 - Debugging and Analyzing Correctness of CUDA Applications
  - DLIT41276 - Developer Tools Fundamentals for Ray Tracing using NVIDIA Nsight Graphics and NVIDIA Nsight Systems
- Ampere Architecture Detailed Blog
  - NVIDIA Ampere Architecture In-Depth
- Developer Tools are free and packaged in the latest version of the CUDA Toolkit
  - https://developer.nvidia.com/cuda-downloads
- Support is available via:
  - https://forums.developer.nvidia.com/c/development-tools/
- More information at:
  - https://developer.nvidia.com/tools-overview

HANDS-ON

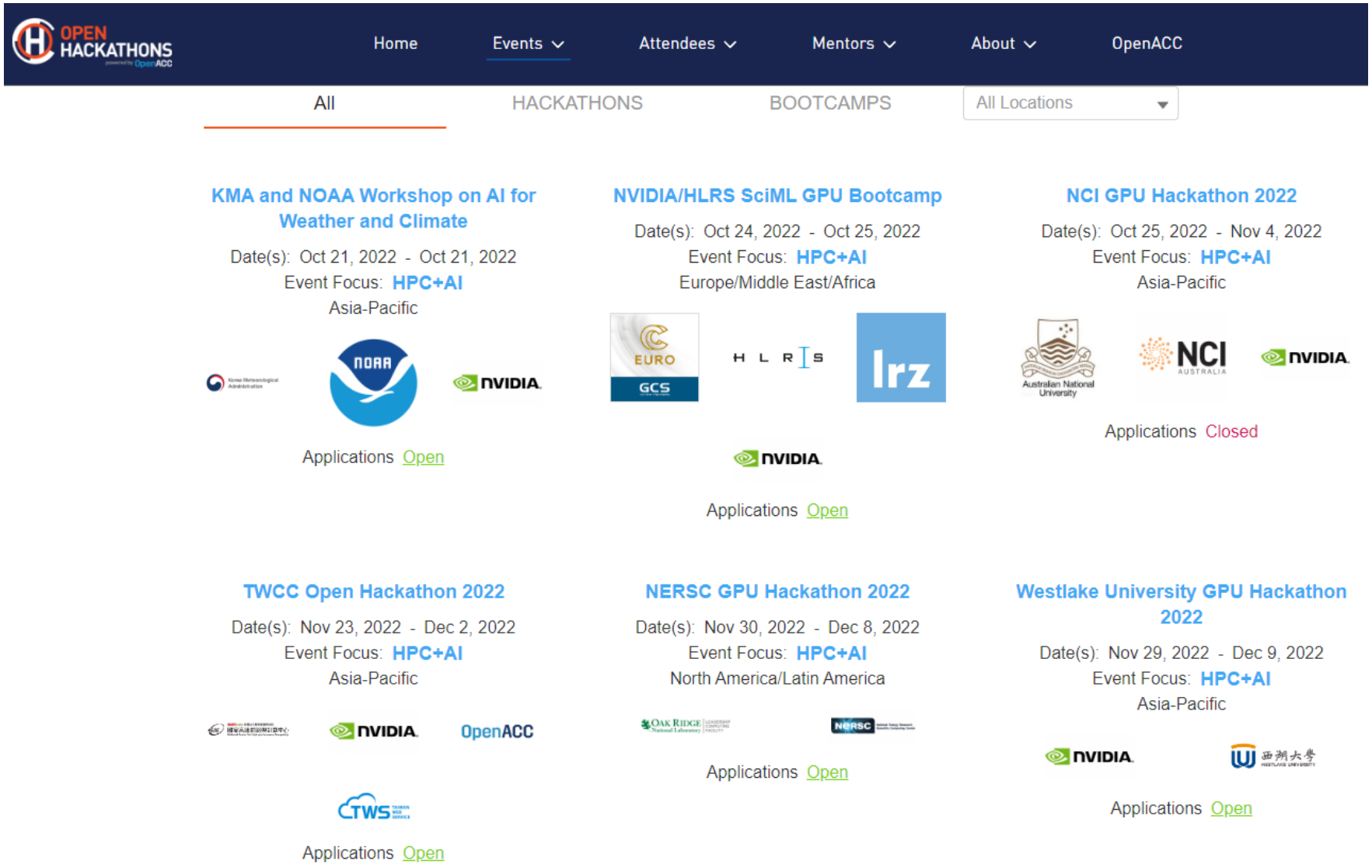
/lus/eagle/projects/SDL_Workshop/jacobi

- Solving Laplace Equation with Jacobi Iterations

$$f_{i,j} = (f_{i+1,j} + f_{i-1,j} + f_{i,j+1}, f_{i,j-1}) / 4$$

```
// set initial conditions at f[0][:], f[N-1][:], f[:][0], f[:][N-1]

while (error > tolerance):
    error = 0
    for i = 1, N-2:
        for j = 1, N-2:
            f[i][j] = 0.5 * (f_old[i+1][j] + f_old[i-1][j] + f_old[i][j+1] + f_old[i][j-1])
            error += (f[i][j] - f_old[i][j]) * (f[i][j] - f_old[i][j])
    swap(f_old, f)
```

WWW.OPENHACKATHONS.ORG




NVIDIA