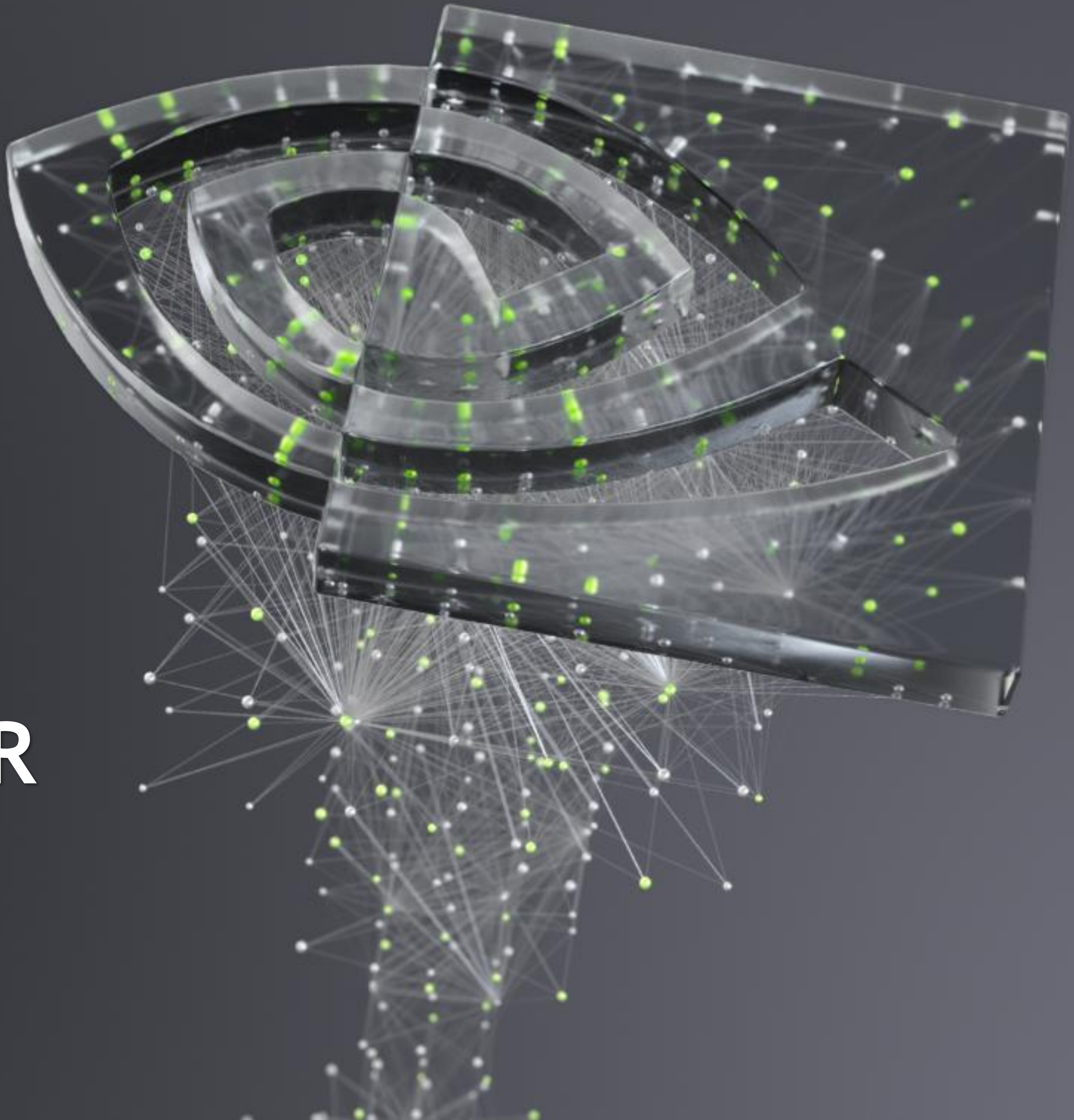
WHAT THE PROFILER IS TELLING YOU

Mozhgan Kabiri chimeh, Sept 2020



BEFORE YOU START

Steps to enlightenment

- Know your application
  - What does it compute? How is it parallelized? What final performance is expected?
- Know your hardware
  - What are the target machines and how many? Machine-specific optimizations okay?
- Know your tools
  - Strengths and weaknesses of each tool? Learn how to use them.
- Know your process
  - Performance optimization is a constant learning process.

Outline

1. Overview of the tools
2. Demo
3. Example application
4. Optimization

NVIDIA NSIGHT FAMILY

https://developer.nvidia.com/blog/migrating-nvidia-nsight-tools-nvvp-nvprof/

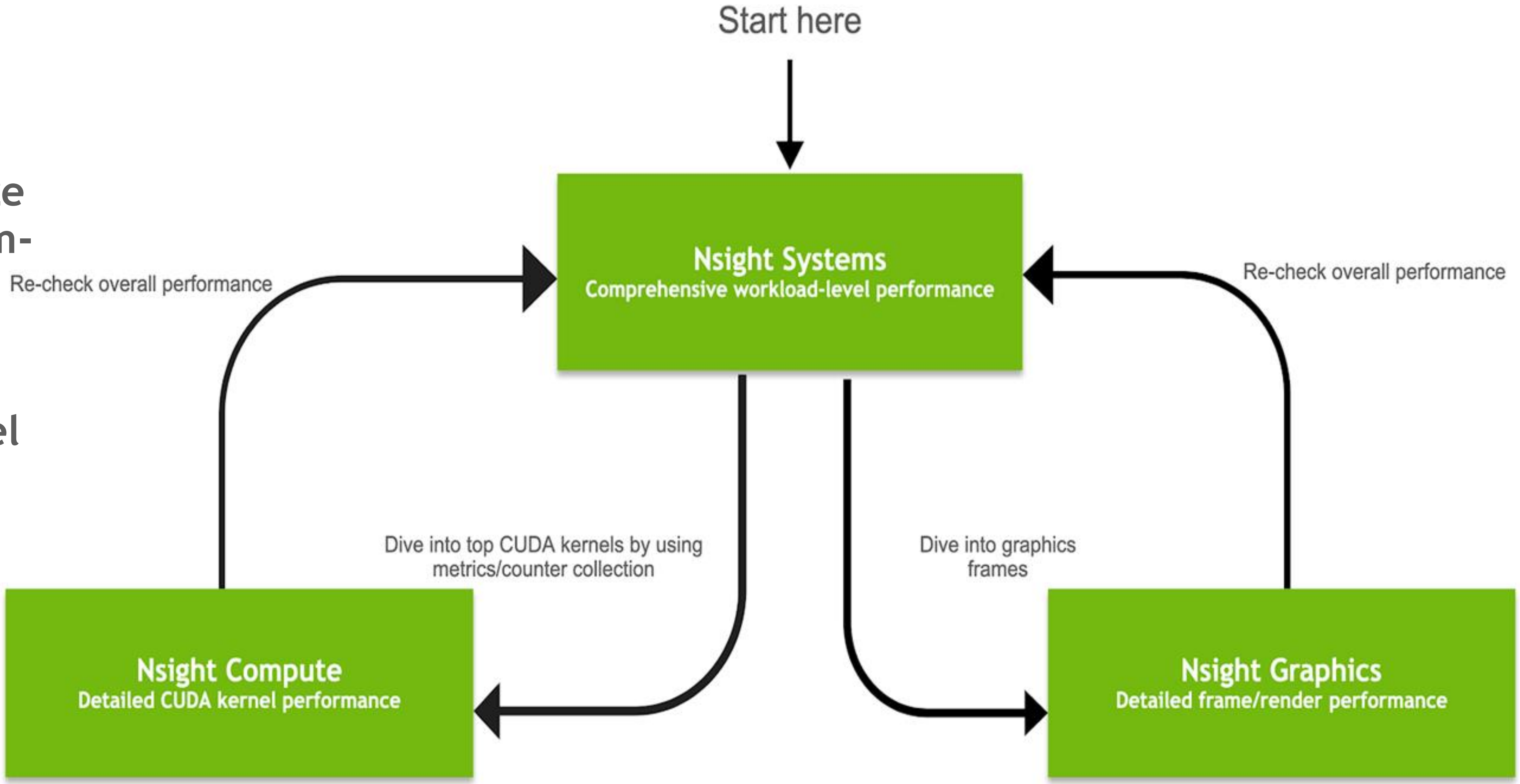
Nsight Product Family

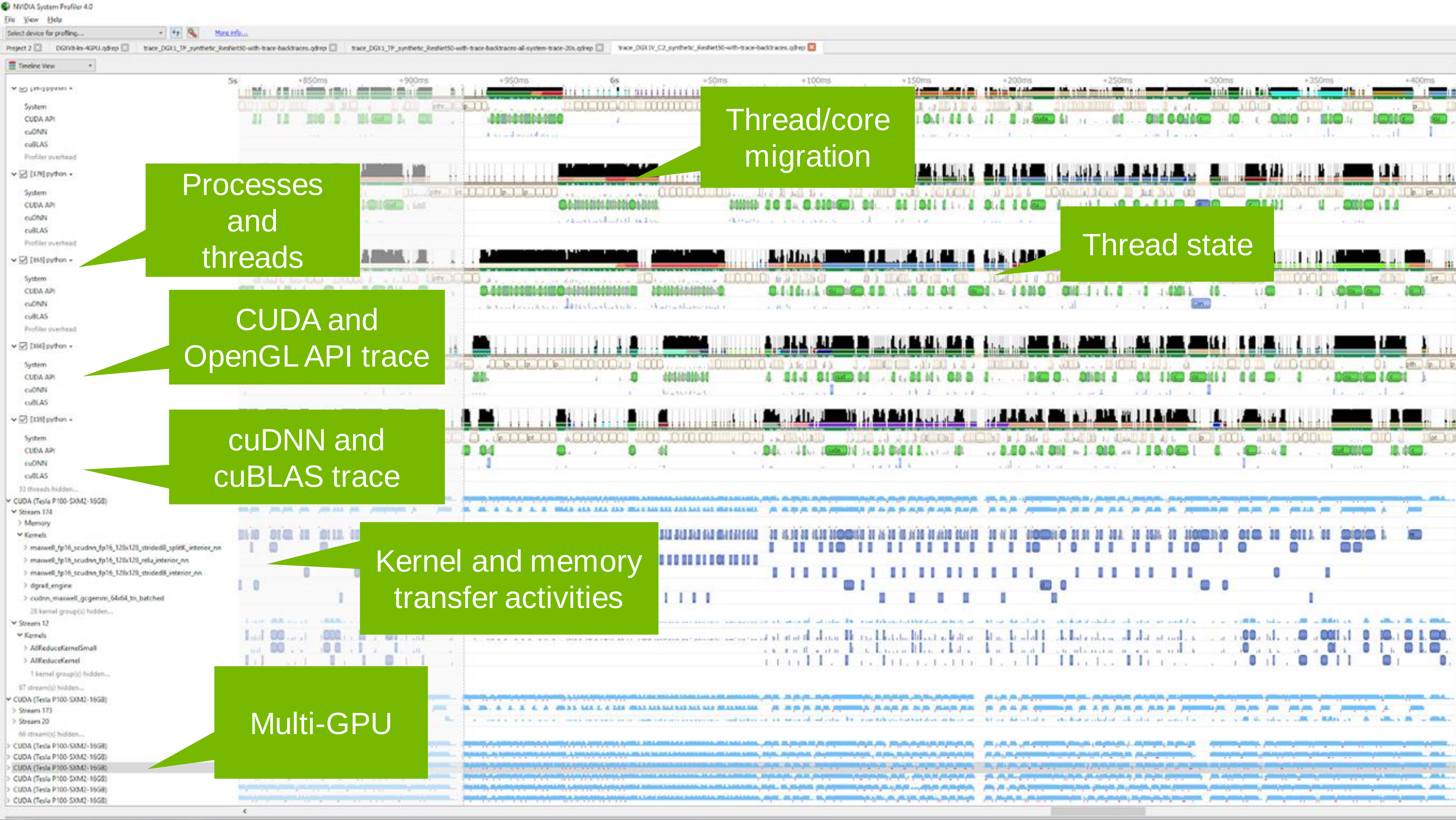
Workflow

Nsight Systems - Analyze application algorithm system-wide

Nsight Compute - Debug/optimize CUDA kernel

Nsight Graphics - Debug/optimize graphics workloads


Start here
Nsight Systems
Comprehensive workload-level performance
Re-check overall performance
Re-check overall performance
Dive into top CUDA kernels by using metrics/counter collection
Dive into graphics frames
Nsight Compute
Detailed CUDA kernel performance
Nsight Graphics
Detailed frame/render performance


NVIDIA System Profiler 4.0
File View Help
Select device for profiling...
More info...
Project 2
DGXV8-lm-4GPU.qdrep
trace_DGX1_TF_synthetic_ResNet50-with-trace-backtraces.qdrep
trace_DGX1_TF_synthetic_ResNet50-with-trace-backtraces-all-system-trace-20s.qdrep
trace_DGX1V_C2_synthetic_ResNet50-with-trace-backtraces.qdrep
Timeline View
5s
+850ms
+900ms
+950ms
6s
+50ms
+100ms
+150ms
+200ms
+250ms
+300ms
+350ms
+400ms
System
CUDA API
cuDNN
cuBLAS
Profiler overhead
[178] python
[165] python
[166] python
[159] python
33 threads hidden...
CUDA (Tesla P100-SXM2-16GB)
Stream 174
Memory
Kernels
maxwell_fp16_scudnn_fp16_128x128_stridedB_splitK_interior_nn
maxwell_fp16_scudnn_fp16_128x128_relu_interior_nn
maxwell_fp16_scudnn_fp16_128x128_stridedB_interior_nn
dgrad_engine
cudnn_maxwell_gcgemm_64x64_tn_batched
28 kernel group(s) hidden...
Stream 12
AllReduceKernelSmall
AllReduceKernel
1 kernel group(s) hidden...
67 stream(s) hidden...
Stream 173
Stream 20
66 stream(s) hidden...

Thread/core migration

Processes and threads

Thread state

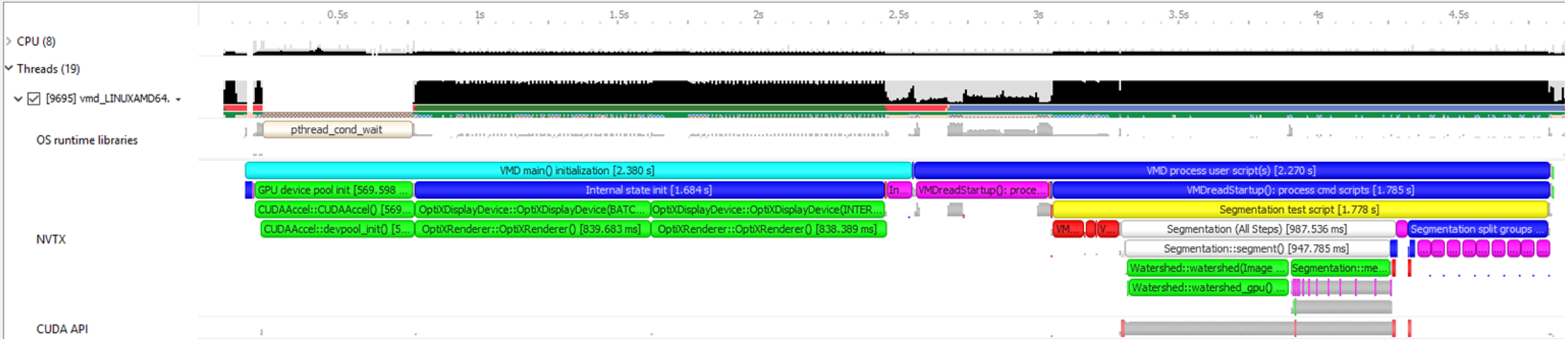
CUDA and OpenGL API trace

cuDNN and cuBLAS trace

Kernel and memory transfer activities

Multi-GPU

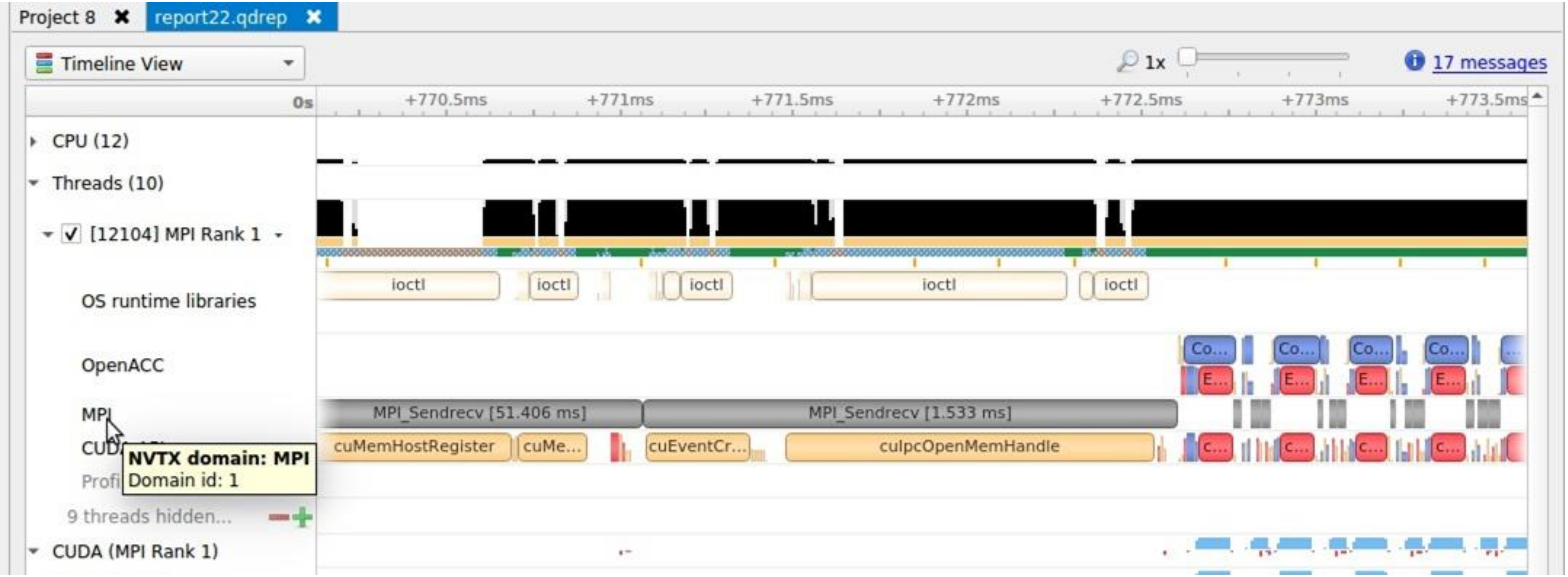
DEMO


0.5s
1s
1.5s
2s
2.5s
3s
3.5s
4s
4.5s
CPU (8)
Threads (19)
☑ [9695] vmd_LINUXAMD64.
OS runtime libraries
pthread_cond_wait
NVTX
VMD main() initialization [2.380 s]
VMD process user script(s) [2.270 s]
GPU device pool init [569.598 ...
Internal state init [1.684 s]
In...
VMDreadStartup(): proce...
VMDreadStartup(): process cmd scripts [1.785 s]
CUDAAccel::CUDAAccel() [569...
OptiXDisplayDevice::OptiXDisplayDevice(BATC...
OptiXDisplayDevice::OptiXDisplayDevice(INTER...
Segmentation test script [1.778 s]
CUDAAccel::devpool_init() [5...
OptiXRenderer::OptiXRenderer() [839.683 ms]
OptiXRenderer::OptiXRenderer() [838.389 ms]
VM...
V...
Segmentation (All Steps) [987.536 ms]
Segmentation split groups ...
Segmentation::segment() [947.785 ms]
Watershed::watershed(Image ...
Segmentation::me...
Watershed::watershed_gpu() ...
CUDA API

USER ANNOTATIONS APIS FOR CPU & GPU
NVTX, OPENGL, VULKAN, AND DIRECT3D PERFORMANCE MARKERS

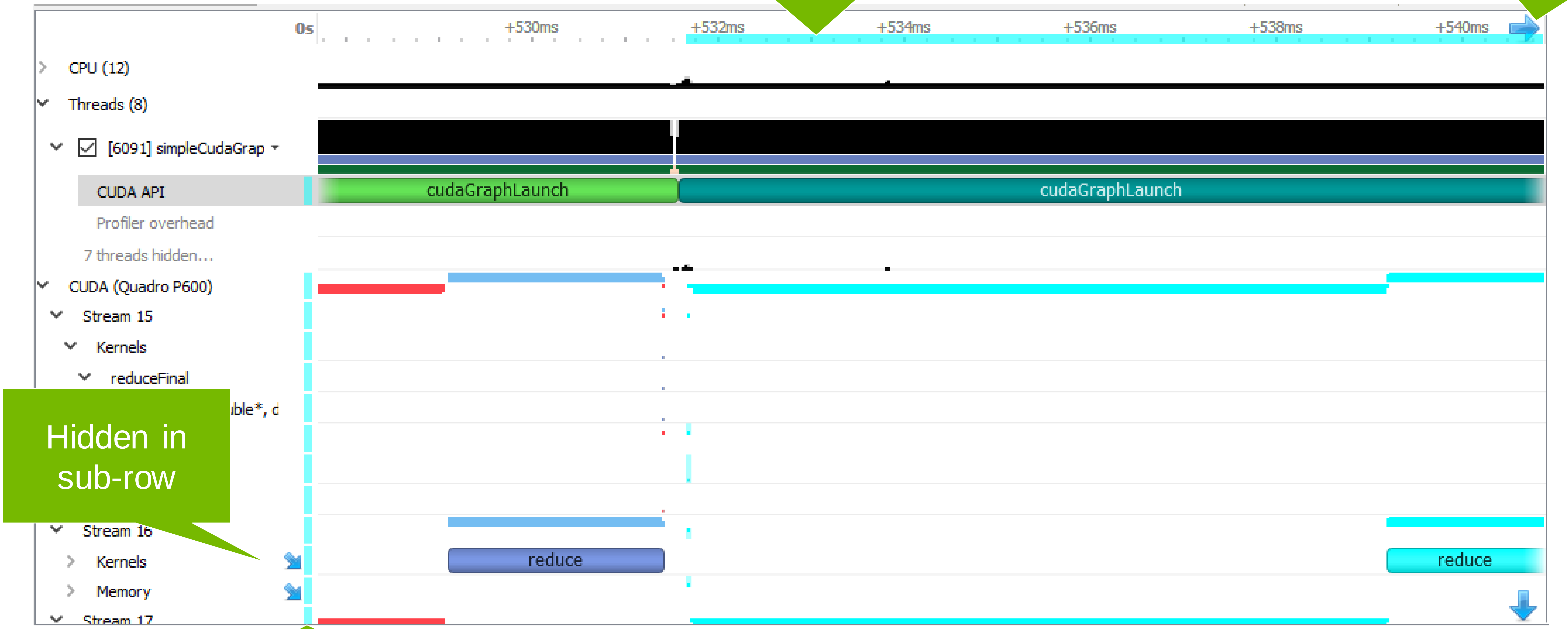
EXAMPLE: VISUAL MOLECULAR DYNAMICS (VMD) ALGORITHMS VISUALIZED WITH NVTX ON CPU

MPI & OPENACC TRACE



NVTX domain: MPI
Domain id: 1

CORRELATION



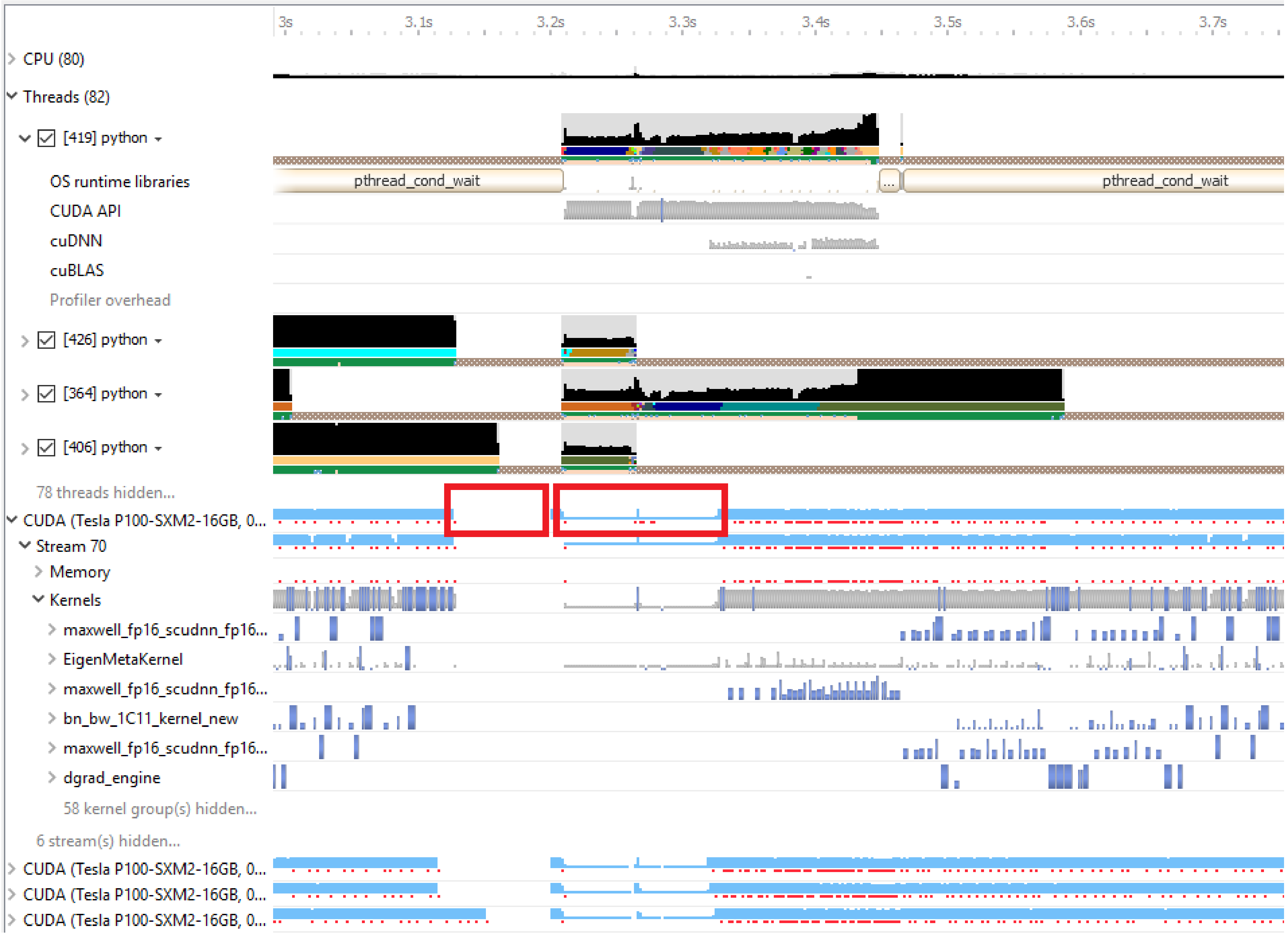
Highlights in ruler

Hidden to right

Hidden in sub-row

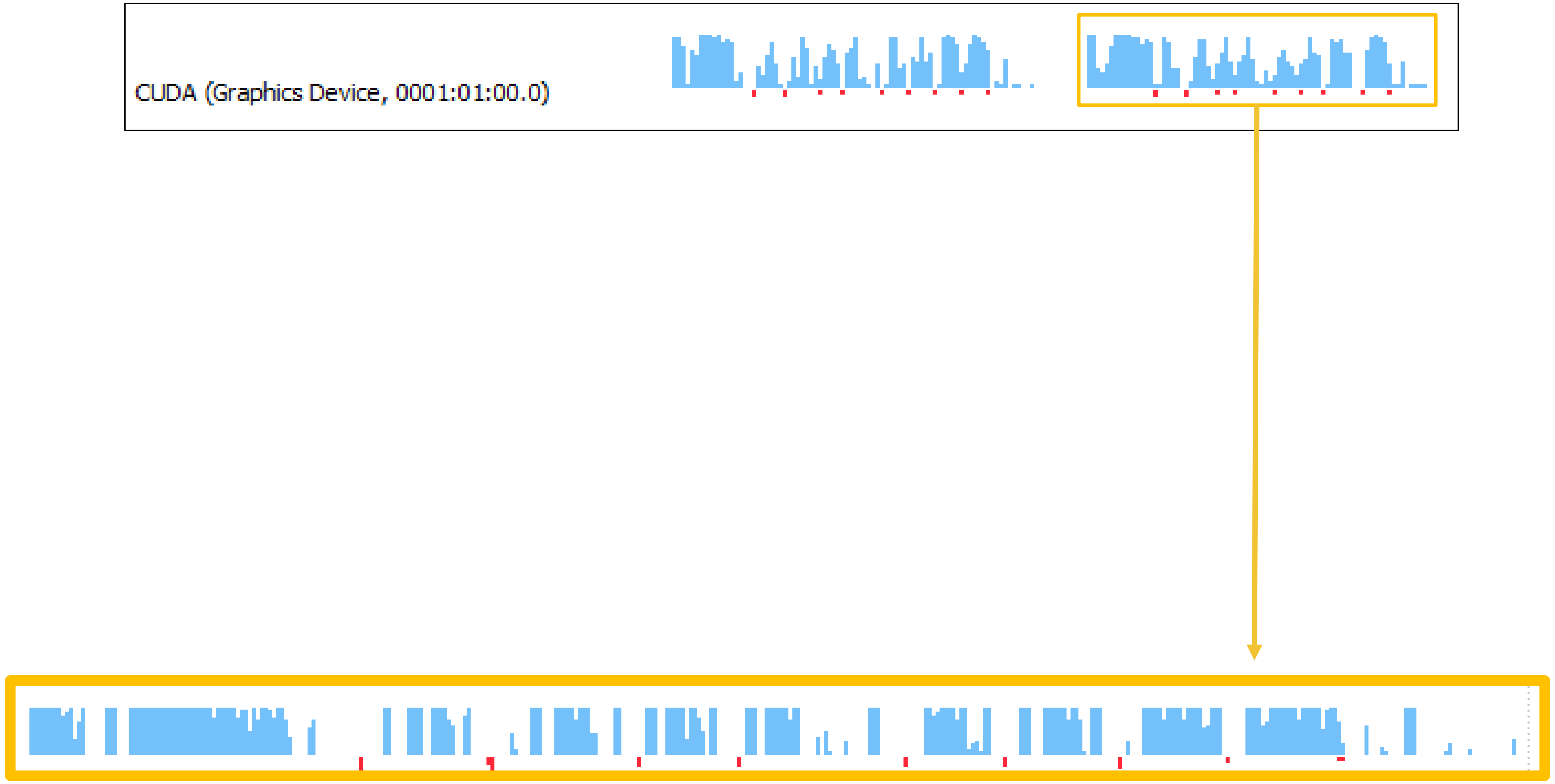
Row has highlights

Hidden below



GPU IDLE AND LOW UTILIZATION LEVEL OF DETAIL

GPU UTILIZATION BASED ON PERCENTAGE TIME COVERAGE



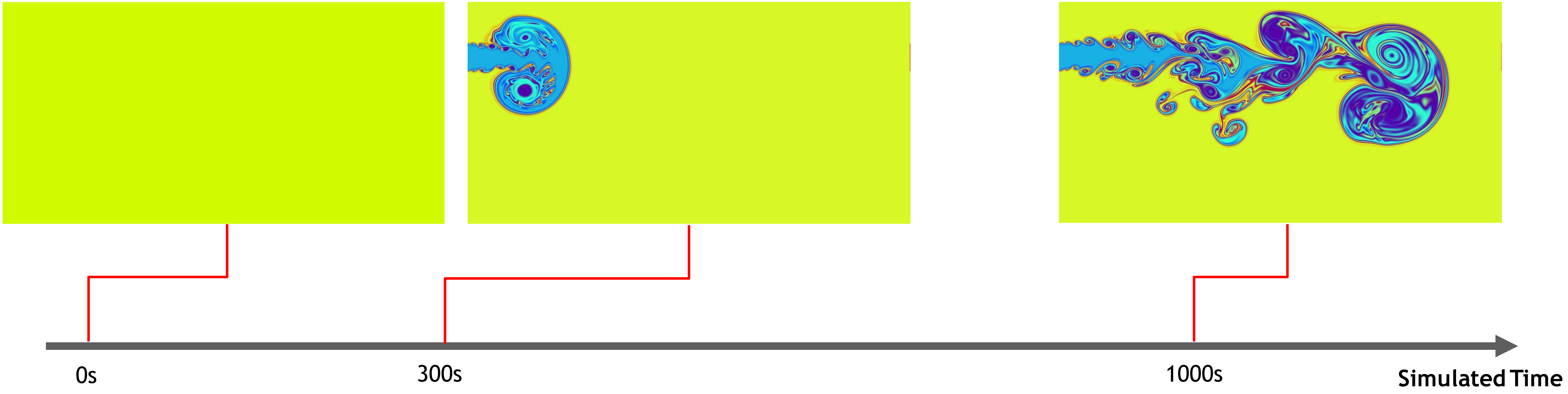
ZOOMING IN REVEALS GAPS WHERE THERE WERE VALLEYS

EXAMPLE APPLICATION

A SAMPLE OF A FLUID SIMULATION

In the context of atmosphere and weather simulation[1]

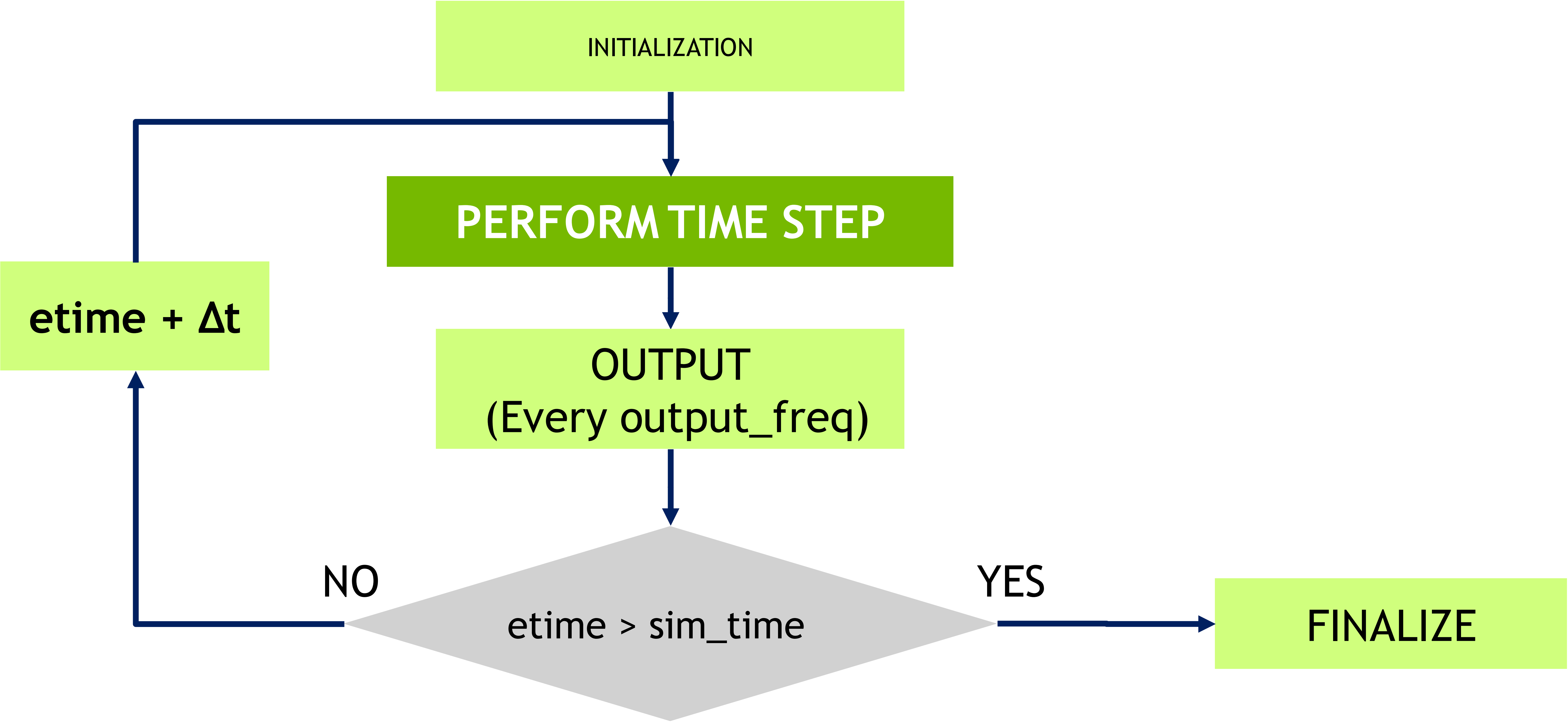
A narrow jet of fast and slightly cold wind is injected into a balanced, neutral atmosphere at rest from the left domain near the model top.



[1]. https://github.com/mrnorman/miniWeather

OUTER LOOP

The main() function

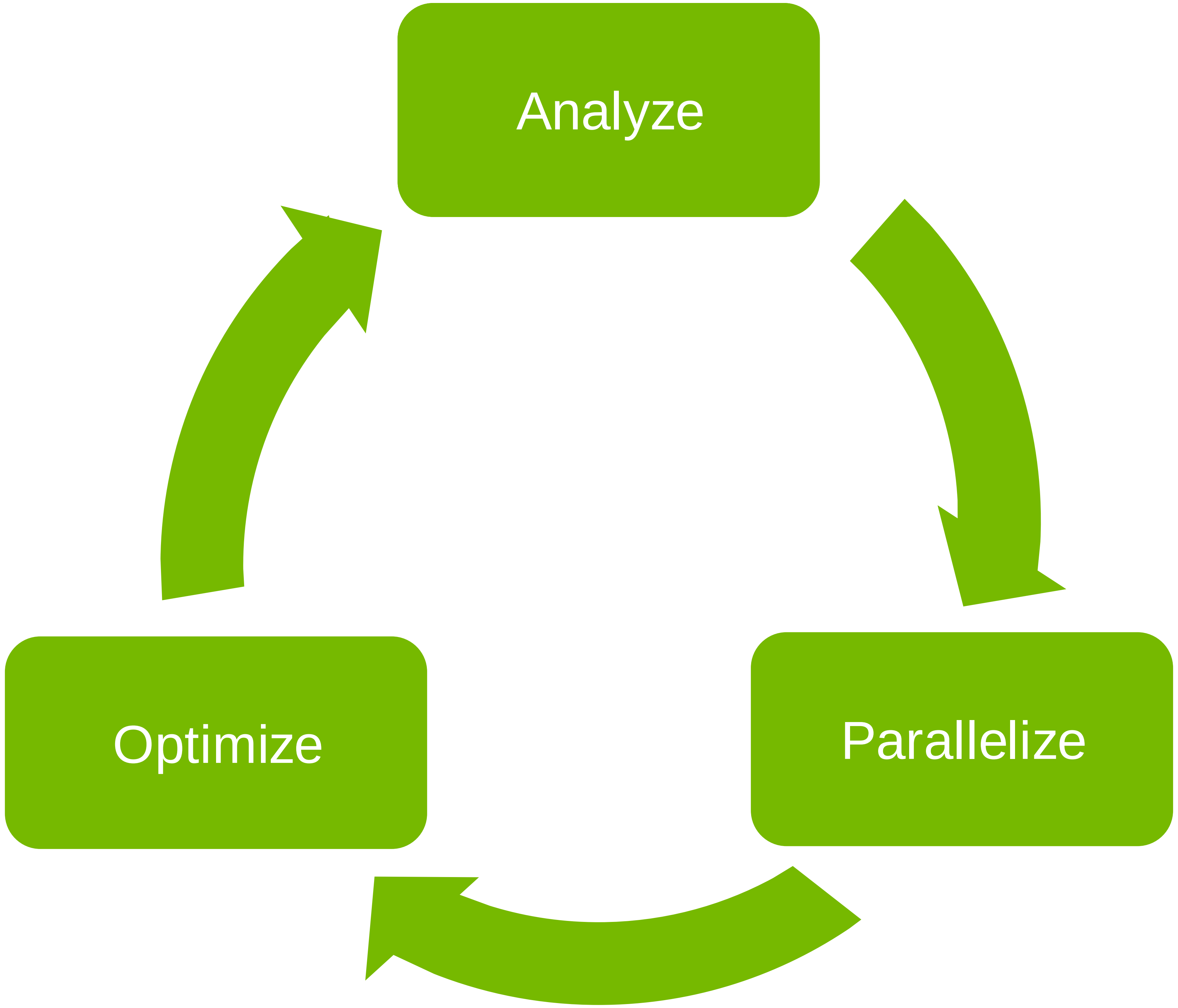


DEVELOPMENT CYCLE

- **Analyze** your code to determine most likely places needing parallelization or optimization.

- **Parallelize** your code by starting with the most time consuming parts and check for correctness.

- **Optimize** your code to improve observed speed-up from parallelization.



TOOLS WE WILL USE: NSIGHT SUITE

Application-wide profiling (Systems), Kernel-level profiling (Compute)

Instrument with NVIDIA Tools Extension (NVTX): Automatic or manual

- Create (nested) ranges, define macros
- Compiler instrumentation

Tracing: CUDA API calls, NVTX trace

Sampling, hardware counters

Nsight Systems

Nsight Compute

NVTX primer: https://devblogs.nvidia.com/parallelforall/cuda-pro-tip-generate-custom-application-profile-timelines-nvtx/

A FIRST (I)NSIGHT

Maximum achievable speedup: Amdahl‘s law

Amdahl‘s law states overall speedup ***s*** given the parallel fraction p of code and number of processes N

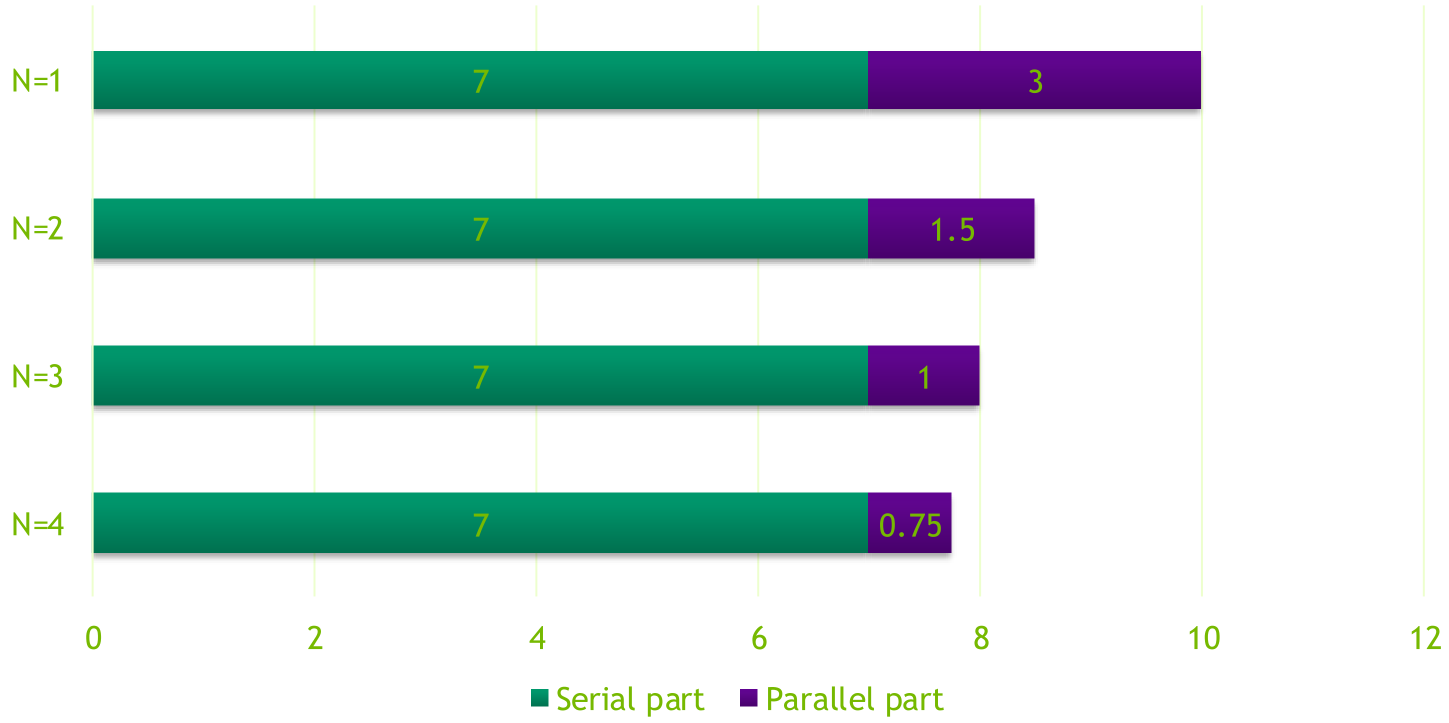
$$\boldsymbol{s} = \frac{1}{1 - p + \frac{p}{N}} < \frac{1}{1 - p}$$

Limited by serial fraction, even for $N \to \infty$

Example for $p = 30\%$

Also valid for per-method speedups

Using 1 to 4 processes



A FIRST (I)NSIGHT

Recording an application timeline

1) Recording, via the GUI (not shown here) or via command line

```
nsys profile -t nvtx,openacc --stats=true --force-overwrite true -o
my_report ./myapp
```

- `profile` - start a profiling session
- `-t`: Selects the APIs to be traced (nvtx and openacc in this example)
- `--stats`: if true, it generates summary of statistics after the collection
- `--force-overwrite`: if true, it overwrites the existing generated report
- `-o` - name for the intermediate result file, created at the end of the collection (`.qdrep` filename)

`nsys --help or nsys [specific command] --help`

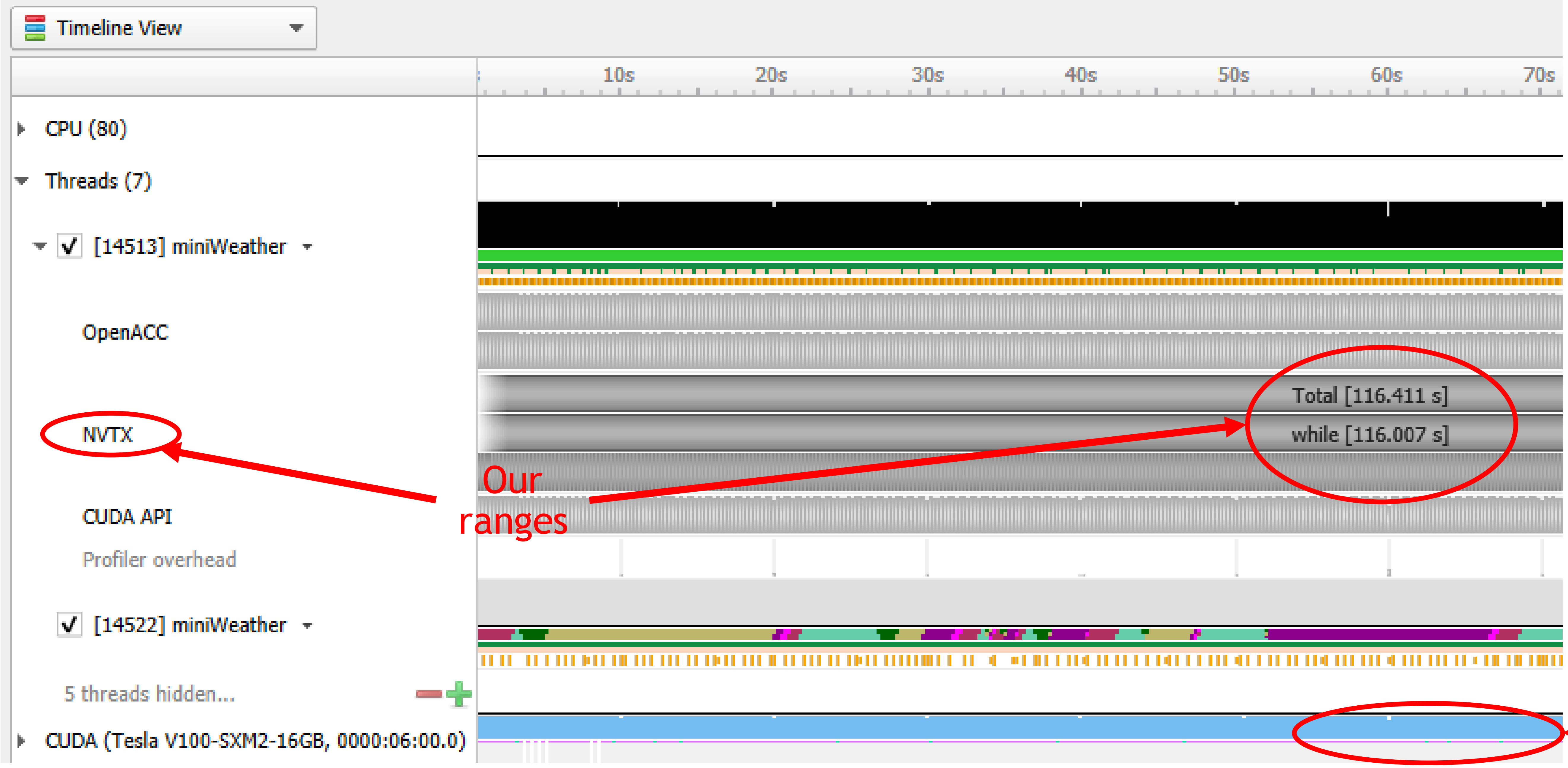
2) Inspect results: Open the report file in the GUI

Also possible to get details on command line (documentation)

See also https://docs.nvidia.com/nsight-systems/, "Profiling from the CLI on Linux Devices"

A FIRST (I)NSIGHT

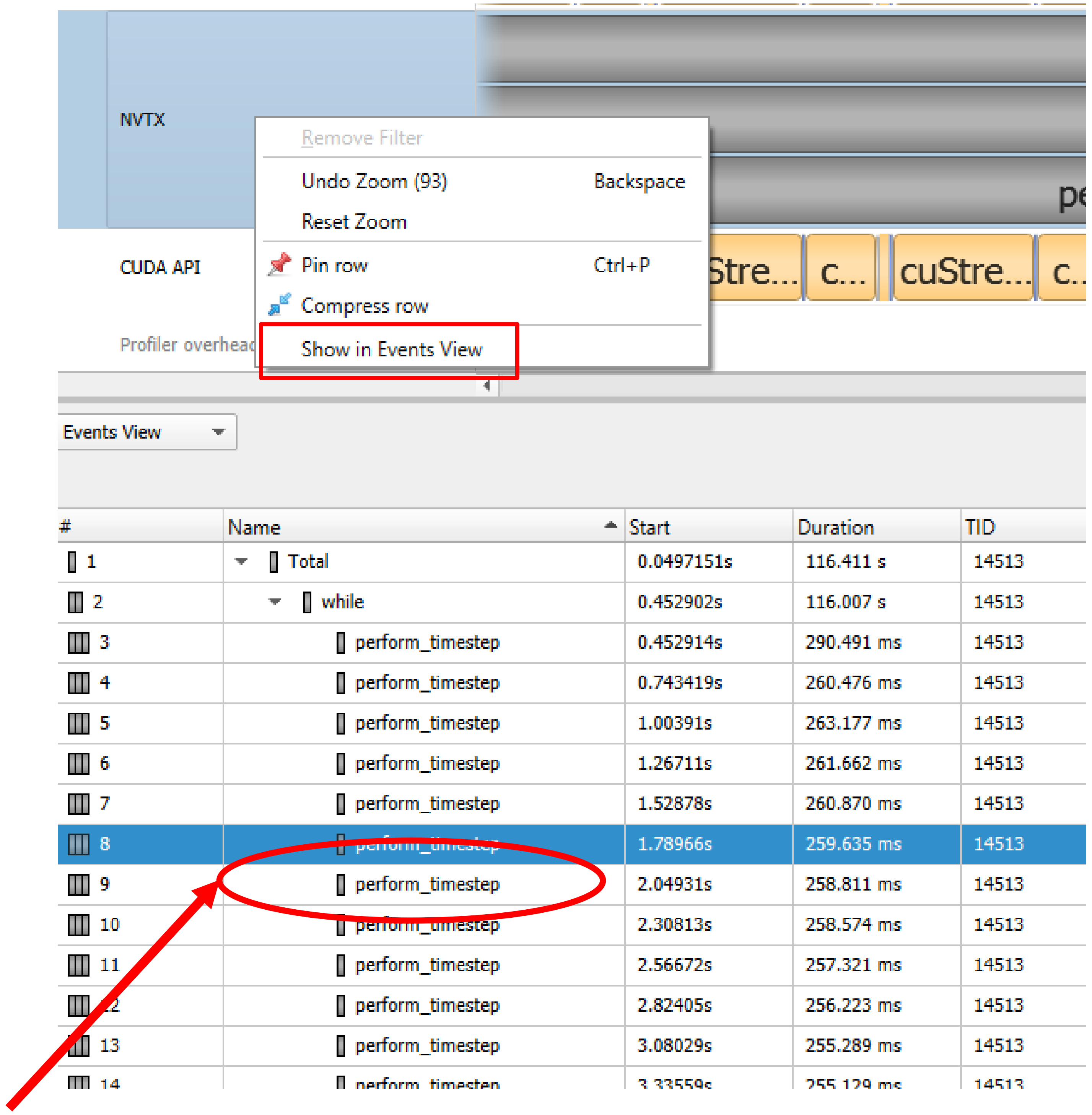
Timeline overview in Nsight Systems GUI



Application already ported to GPU
- basic guidelines followed

S7122: CUDA Optimization Tips, Tricks and Techniques (2017)

LOOKING CLOSER



#	Name	Start	Duration	TID
1	Total	0.0497151s	116.411 s	14513
2	while	0.452902s	116.007 s	14513
3	perform_timestep	0.452914s	290.491 ms	14513
4	perform_timestep	0.743419s	260.476 ms	14513
5	perform_timestep	1.00391s	263.177 ms	14513
6	perform_timestep	1.26711s	261.662 ms	14513
7	perform_timestep	1.52878s	260.870 ms	14513
8	perform_timestep	1.78966s	259.635 ms	14513
9	perform_timestep	2.04931s	258.811 ms	14513
10	perform_timestep	2.30813s	258.574 ms	14513
11	perform_timestep	2.56672s	257.321 ms	14513
12	perform_timestep	2.82405s	256.223 ms	14513
13	perform_timestep	3.08029s	255.289 ms	14513
14	perform_timestep	3.33559s	255.129 ms	14513

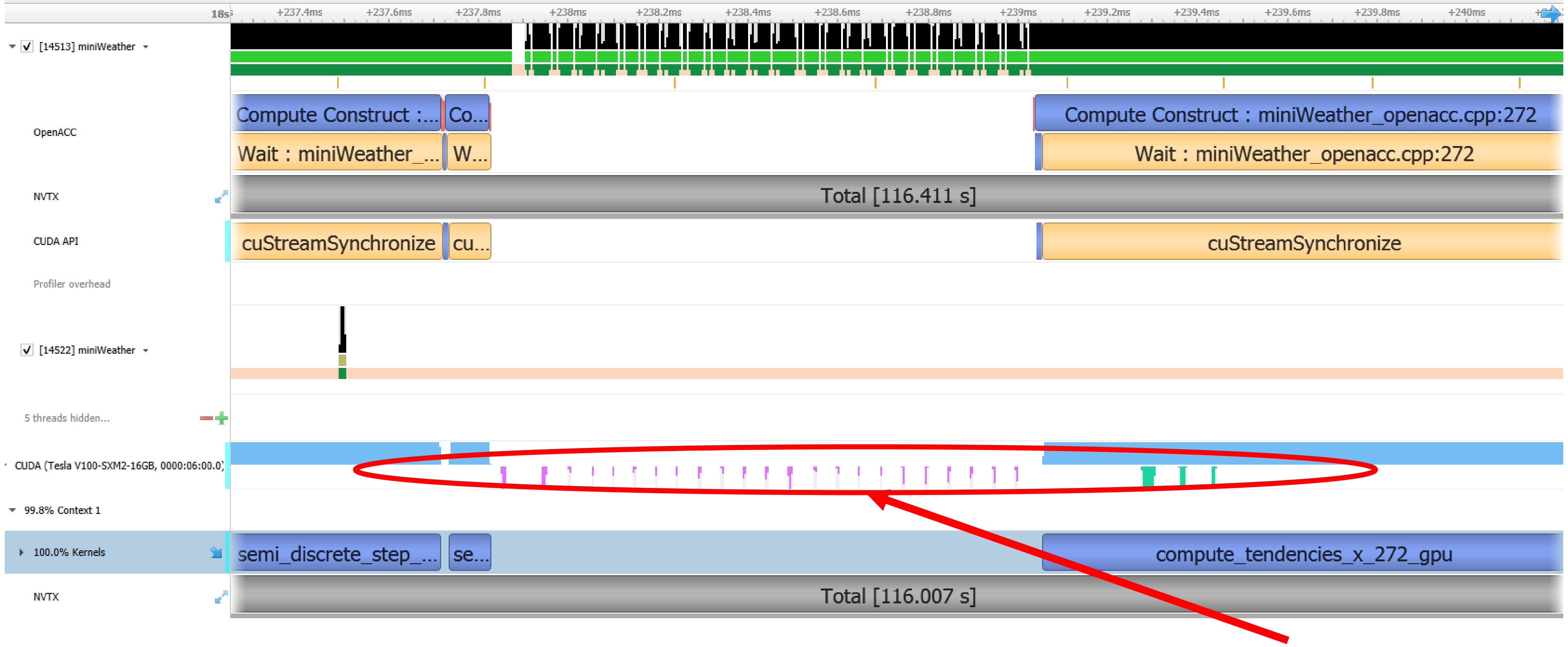
Our focus

Zooming in and using Events View for NVTX

Useful for other rows, e.g. CUDA API

Hierarchy of ranges, use to locate on timeline

LOOKING CLOSER



Data transfer

Learn more about Unified Memory:

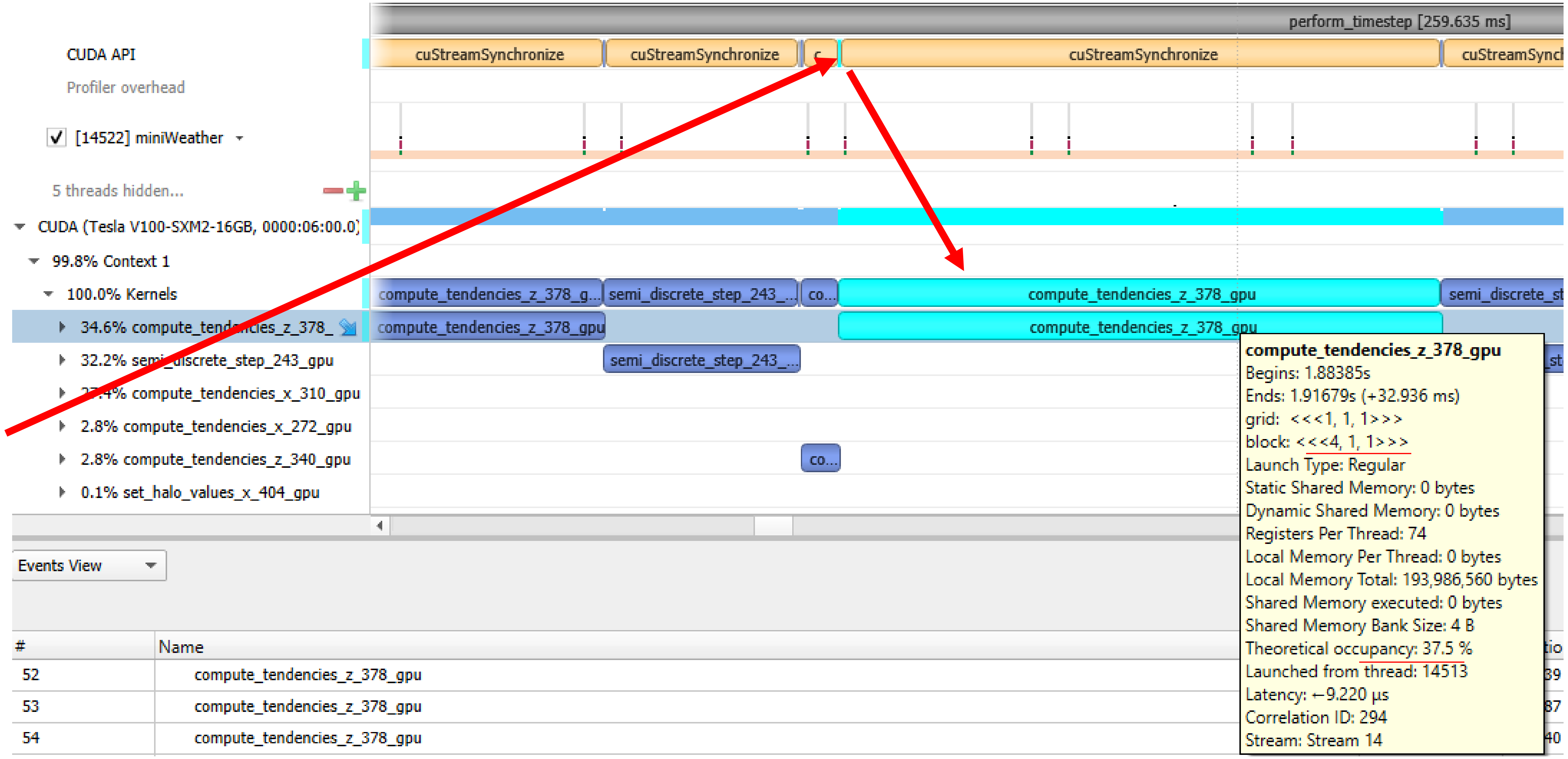
S8430: Everything You Need to Know About Unified Memory (2018)

S9727: Memory Management on Modern GPU Architectures (2019)

IDENTIFYING INTERESTING REGIONS

How to correlate ranges, API and kernel calls

Identify components. Mark kernel in CUDA API row, find kernel launch


perform_timestep [259.635 ms]
CUDA API
cuStreamSynchronize
cuStreamSynchronize
c...
cuStreamSynchronize
cuStreamSyncl
Profiler overhead
☑ [14522] miniWeather
5 threads hidden...
CUDA (Tesla V100-SXM2-16GB, 0000:06:00.0)
99.8% Context 1
100.0% Kernels
compute_tendencies_z_378_g...
semi_discrete_step_243_...
co...
compute_tendencies_z_378_gpu
semi_discrete_st
34.6% compute_tendencies_z_378_
compute_tendencies_z_378_gpu
compute_tendencies_z_378_gpu
32.2% semi_discrete_step_243_gpu
semi_discrete_step_243_...
27.4% compute_tendencies_x_310_gpu
2.8% compute_tendencies_x_272_gpu
2.8% compute_tendencies_z_340_gpu
co...
0.1% set_halo_values_x_404_gpu
Events View

compute_tendencies_z_378_gpu
Begins: 1.88385s
Ends: 1.91679s (+32.936 ms)
grid: <<<1, 1, 1>>>
block: <<<4, 1, 1>>>
Launch Type: Regular
Static Shared Memory: 0 bytes
Dynamic Shared Memory: 0 bytes
Registers Per Thread: 74
Local Memory Per Thread: 0 bytes
Local Memory Total: 193,986,560 bytes
Shared Memory executed: 0 bytes
Shared Memory Bank Size: 4 B
Theoretical occupancy: 37.5 %
Launched from thread: 14513
Latency: ←9.220 μs
Correlation ID: 294
Stream: Stream 14

#	Name	tio
52	compute_tendencies_z_378_gpu	39
53	compute_tendencies_z_378_gpu	87
54	compute_tendencies_z_378_gpu	40

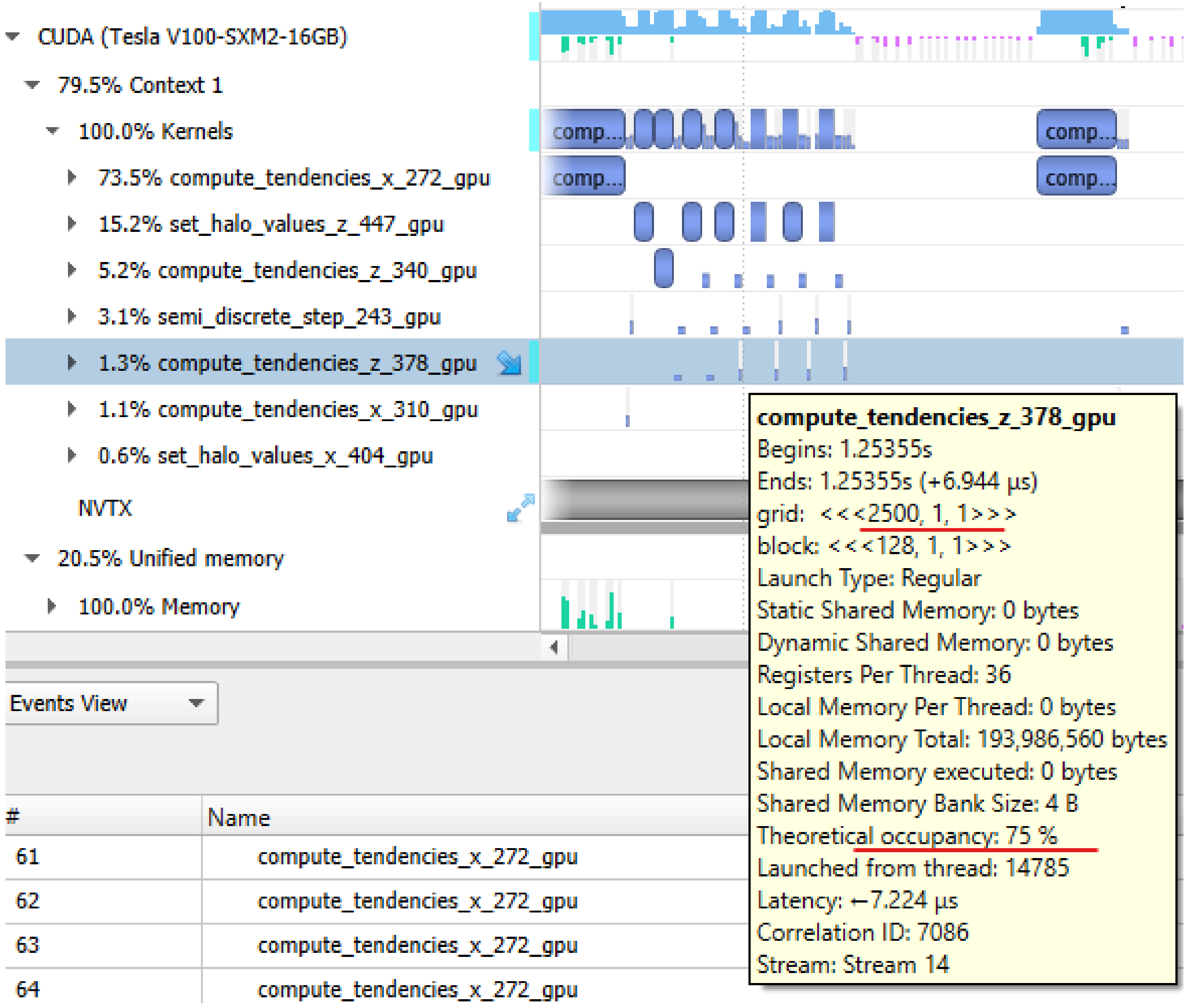
Finding correlations

BUILDING A HYPOTHESIS

```
#pragma acc parallel loop private(indt, indf1, indf2)
 for (ll = 0; ll < NUM_VARS; ll++) {
  for (k = 0; k < nz; k++) {
   for (i = 0; i < nx; i++)  {
    indt = ll * nz * nx + k * nx + i;
    indf1 = ll * (nz + 1) * (nx + 1) + k * (nx + 1) + i;
    indf2 = ll * (nz + 1) * (nx + 1) + k * (nx + 1) + i + 1;
    tend[indt] = -(flux[indf2] - flux[indf1]) / dx;
   }  } }
```

Hypothesis: small iteration count for outer loop

Solution: flatten the loop by collapsing the tightly-nested loops

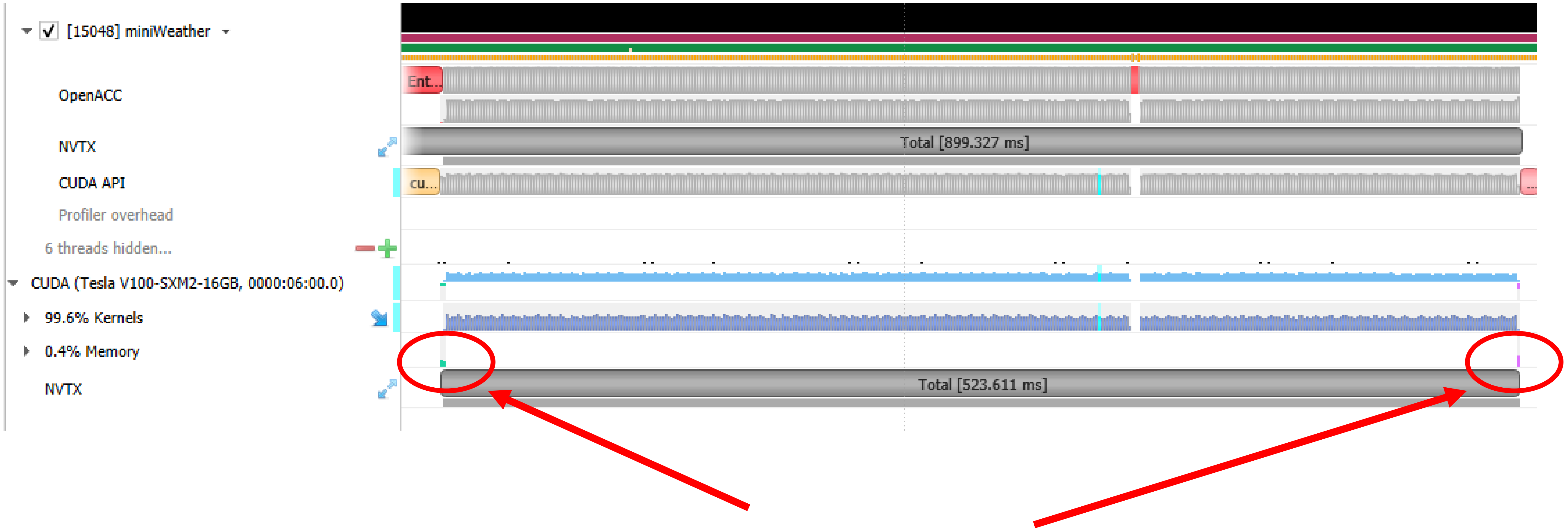


CUDA (Tesla V100-SXM2-16GB)
- 79.5% Context 1
  - 100.0% Kernels
    - 73.5% compute_tendencies_x_272_gpu
    - 15.2% set_halo_values_z_447_gpu
    - 5.2% compute_tendencies_z_340_gpu
    - 3.1% semi_discrete_step_243_gpu
    - 1.3% compute_tendencies_z_378_gpu
    - 1.1% compute_tendencies_x_310_gpu
    - 0.6% set_halo_values_x_404_gpu
  - NVTX
- 20.5% Unified memory
  - 100.0% Memory

#	Name
61	compute_tendencies_x_272_gpu
62	compute_tendencies_x_272_gpu
63	compute_tendencies_x_272_gpu
64	compute_tendencies_x_272_gpu

compute_tendencies_z_378_gpu
Begins: 1.25355s
Ends: 1.25355s (+6.944 μs)
grid: <<<2500, 1, 1>>>
block: <<<128, 1, 1>>>
Launch Type: Regular
Static Shared Memory: 0 bytes
Dynamic Shared Memory: 0 bytes
Registers Per Thread: 36
Local Memory Per Thread: 0 bytes
Local Memory Total: 193,986,560 bytes
Shared Memory executed: 0 bytes
Shared Memory Bank Size: 4 B
Theoretical occupancy: 75 %
Launched from thread: 14785
Latency: ←7.224 μs
Correlation ID: 7086
Stream: Stream 14

LOOKING CLOSER



Data transfers

Learn more about Unified Memory:

S8430: Everything You Need to Know About Unified Memory (2018)

S9727: Memory Management on Modern GPU Architectures (2019)

Result:

Baseline = 116 seconds
Current = 900 ms

RECAP: HIGH-LEVEL ANALYSIS

Application timeline with Nsight Systems

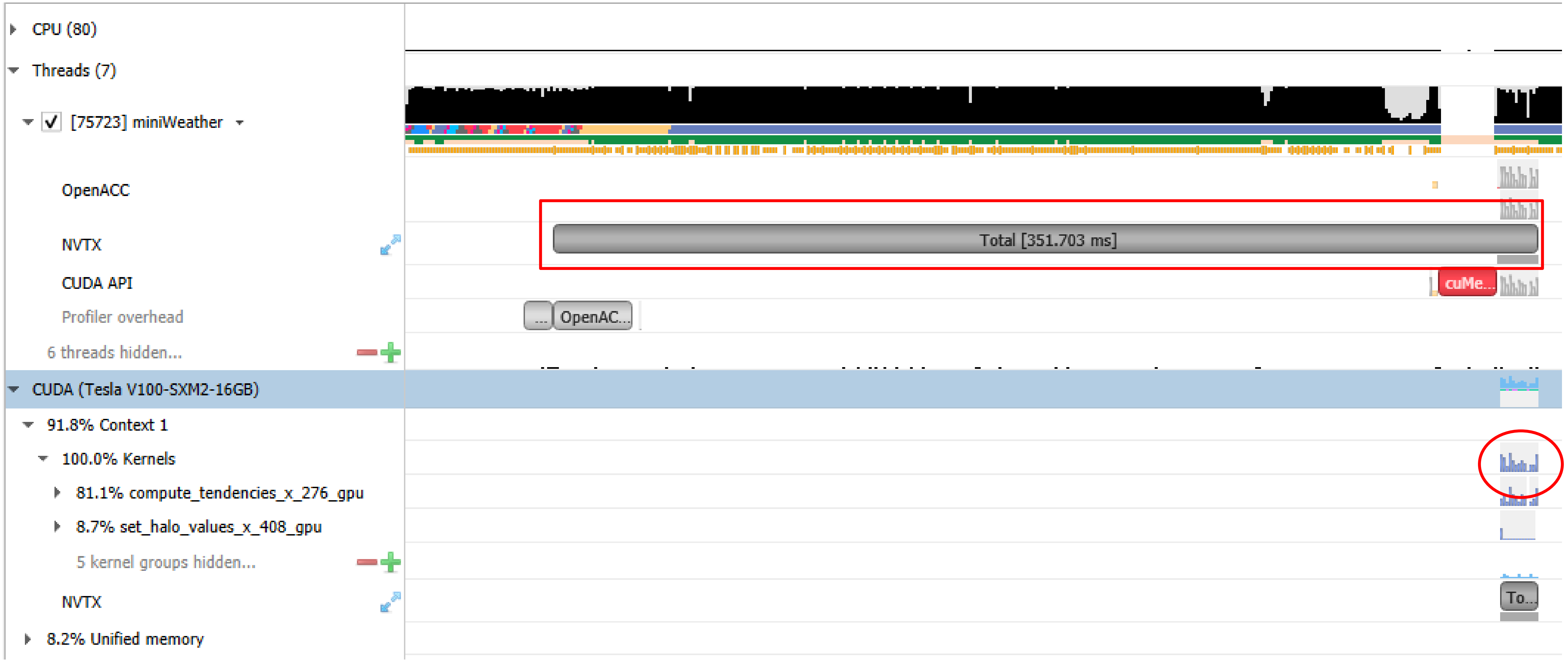
Annotated code

Analyzed regions (NVTX)

Identified first optimization target (Wallclock, Amdahl‘s law)

Correlated with actual kernel launch

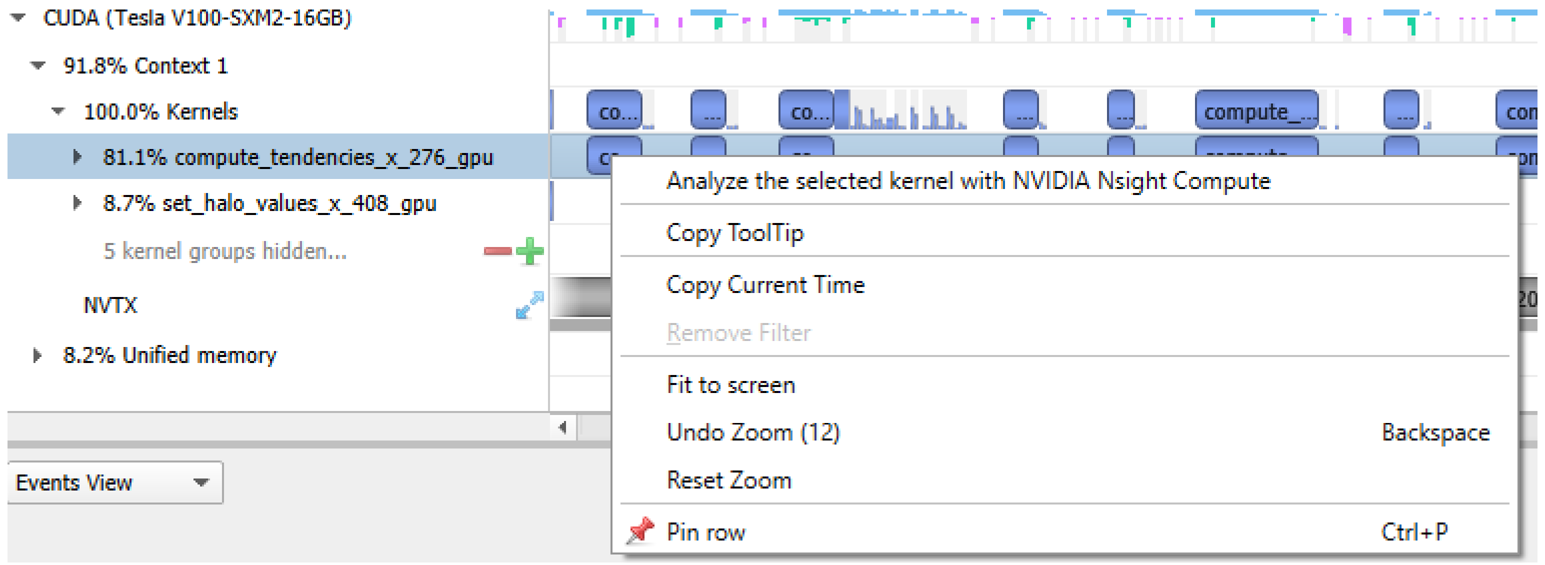
Now: Look briefly at the Nsight Compute



CPU (80)
Threads (7)
[75723] miniWeather
OpenACC
NVTX
CUDA API
Profiler overhead
6 threads hidden...
CUDA (Tesla V100-SXM2-16GB)
91.8% Context 1
100.0% Kernels
81.1% compute_tendencies_x_276_gpu
8.7% set_halo_values_x_408_gpu
5 kernel groups hidden...
NVTX
8.2% Unified memory
Total [351.703 ms]
cuMe...
...
OpenAC...
To...

EXAMPLE ANALYSIS OF A KERNEL

Analysis with Nsight Compute



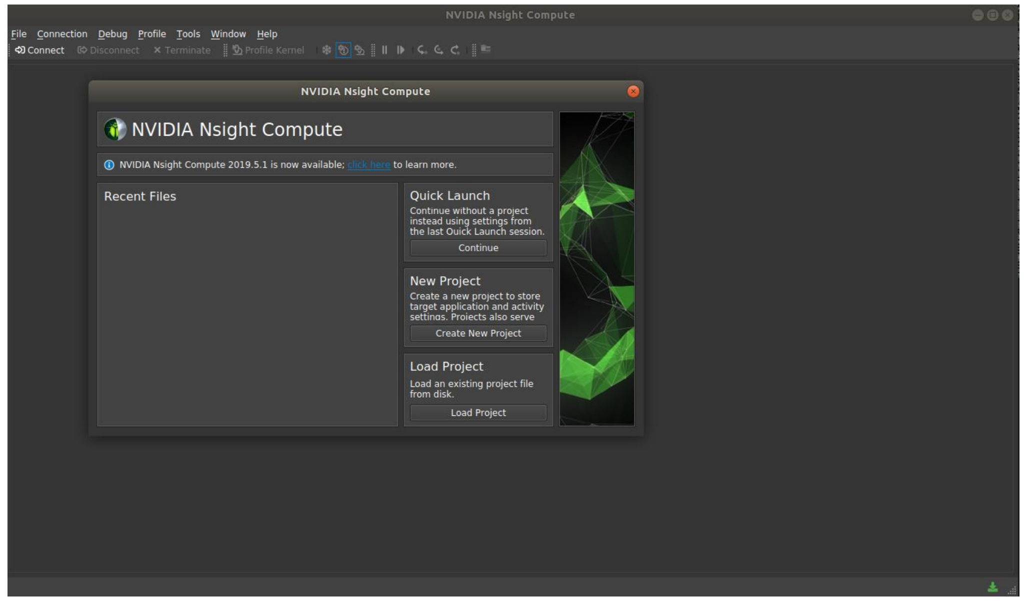
Right-click menu in Nsight Systems, get command line

Run command line

```
ncu --set full -k compute_tendencies_x_276_gpu -s 4 -c 1 -o myreport ./myapp
```

Important switches for metrics collection, pre-selected sets

Fully customizable, `ncu --help`. Check `--list-metrics` and `--query-metrics`

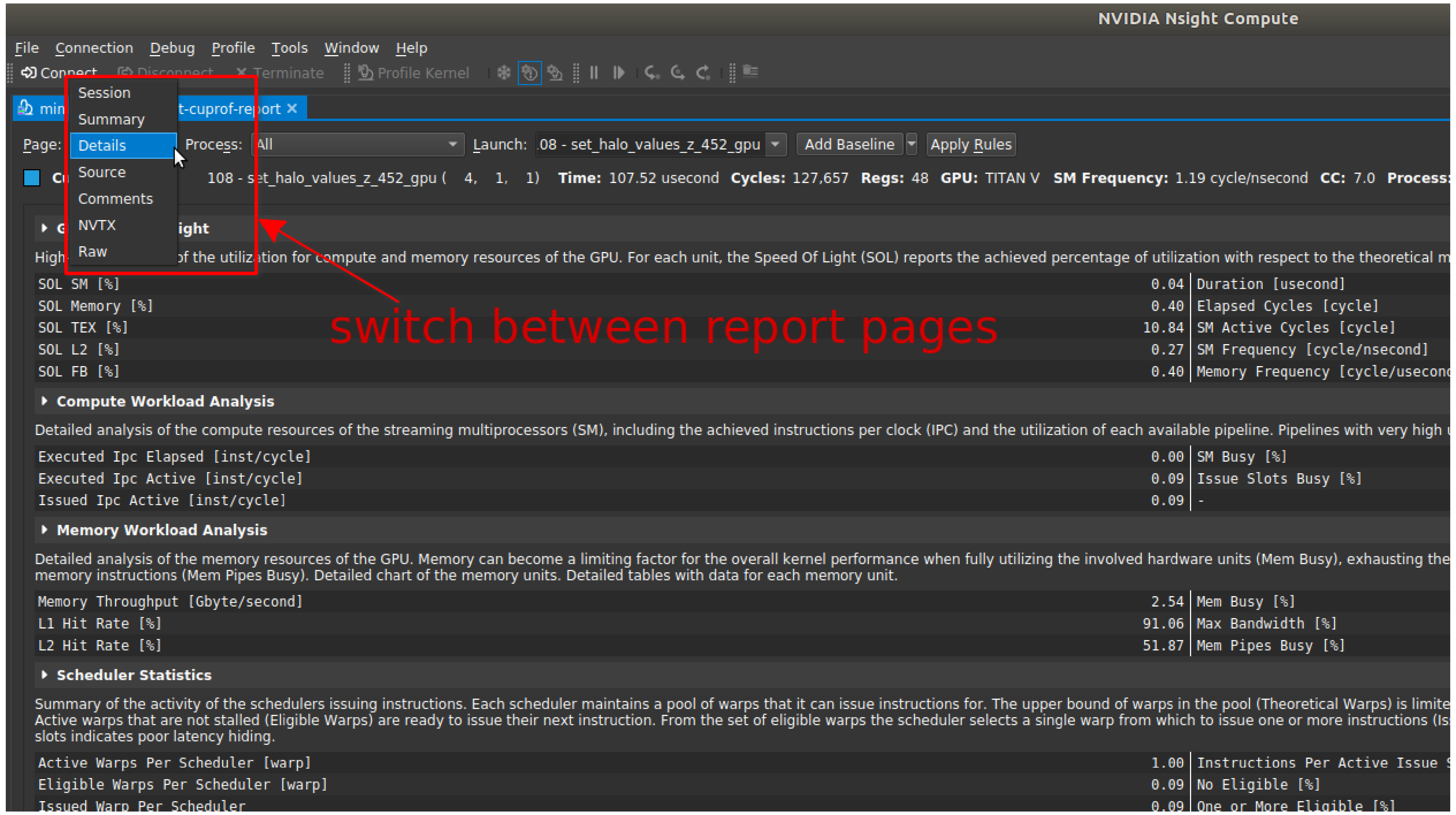


We use GUI for analysis and load report file

Alternatively, interactively run and analyze directly through GUI

See also https://docs.nvidia.com/nsight-compute/, "Nsight Compute CLI"

NSIGHT COMPUTE OVERVIEW



switch between report pages

Page: Details Process: All Launch: 08 - set_halo_values_z_452_gpu

108 - set_halo_values_z_452_gpu (4, 1, 1) **Time:** 107.52 usecond **Cycles:** 127,657 **Regs:** 48 **GPU:** TITAN V **SM Frequency:** 1.19 cycle/nsecond **CC:** 7.0 **Process:**

ight

High of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical m

```
SOL SM [%]                                   0.04 | Duration [usecond]
SOL Memory [%]                               0.40 | Elapsed Cycles [cycle]
SOL TEX [%]                                 10.84 | SM Active Cycles [cycle]
SOL L2 [%]                                   0.27 | SM Frequency [cycle/nsecond]
SOL FB [%]                                   0.40 | Memory Frequency [cycle/usecon
```

Compute Workload Analysis

Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high u

```
Executed Ipc Elapsed [inst/cycle]            0.00 | SM Busy [%]
Executed Ipc Active [inst/cycle]             0.09 | Issue Slots Busy [%]
Issued Ipc Active [inst/cycle]               0.09 | -
```

Memory Workload Analysis

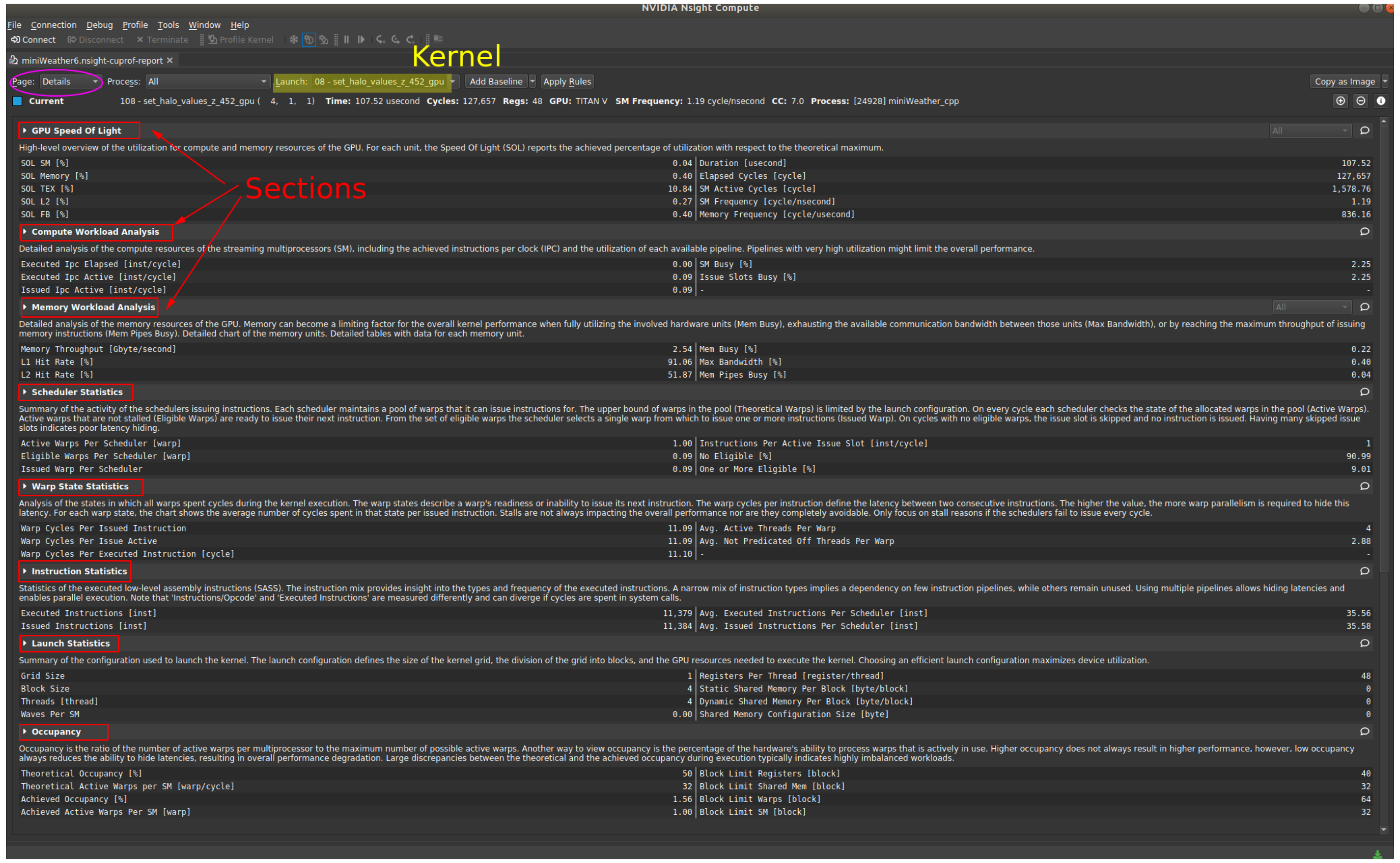
Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.

```
Memory Throughput [Gbyte/second]             2.54 | Mem Busy [%]
L1 Hit Rate [%]                             91.06 | Max Bandwidth [%]
L2 Hit Rate [%]                             51.87 | Mem Pipes Busy [%]
```

Scheduler Statistics

Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limite Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Is slots indicates poor latency hiding.

```
Active Warps Per Scheduler [warp]            1.00 | Instructions Per Active Issue S
Eligible Warps Per Scheduler [warp]          0.09 | No Eligible [%]
Issued Warp Per Scheduler                    0.09 | One or More Eligible [%]
```


NVIDIA Nsight Compute
File Connection Debug Profile Tools Window Help
Connect Disconnect Terminate Profile Kernel
miniWeather6.nsight-cuprof-report
Add Baseline
Apply Rules
Copy as Image
All

Kernel

Sections

Page: Details Process: All Launch: 08 - set_halo_values_z_452_gpu

Current 108 - set_halo_values_z_452_gpu (4, 1, 1) **Time:** 107.52 usecond **Cycles:** 127,657 **Regs:** 48 **GPU:** TITAN V **SM Frequency:** 1.19 cycle/nsecond **CC:** 7.0 **Process:** [24928] miniWeather_cpp

GPU Speed Of Light

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum.

```
SOL SM [%]                                   0.04 | Duration [usecond]                                107.52
SOL Memory [%]                               0.40 | Elapsed Cycles [cycle]                           127,657
SOL TEX [%]                                 10.84 | SM Active Cycles [cycle]                        1,578.76
SOL L2 [%]                                   0.27 | SM Frequency [cycle/nsecond]                        1.19
SOL FB [%]                                   0.40 | Memory Frequency [cycle/usecond]                  836.16
```

Compute Workload Analysis

Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.

```
Executed Ipc Elapsed [inst/cycle]            0.00 | SM Busy [%]                                         2.25
Executed Ipc Active [inst/cycle]             0.09 | Issue Slots Busy [%]                                2.25
Issued Ipc Active [inst/cycle]               0.09 | -                                                      -
```

Memory Workload Analysis

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]             2.54 | Mem Busy [%]                                        0.22
L1 Hit Rate [%]                             91.06 | Max Bandwidth [%]                                   0.40
L2 Hit Rate [%]                             51.87 | Mem Pipes Busy [%]                                  0.04
```

Scheduler Statistics

Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

```
Active Warps Per Scheduler [warp]            1.00 | Instructions Per Active Issue Slot [inst/cycle]        1
Eligible Warps Per Scheduler [warp]          0.09 | No Eligible [%]                                    90.99
Issued Warp Per Scheduler                    0.09 | One or More Eligible [%]                            9.01
```

Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle.

```
Warp Cycles Per Issued Instruction          11.09 | Avg. Active Threads Per Warp                           4
Warp Cycles Per Issue Active                11.09 | Avg. Not Predicated Off Threads Per Warp            2.88
Warp Cycles Per Executed Instruction [cycle] 11.10 | -                                                     -
```

Instruction Statistics

Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.

```
Executed Instructions [inst]               11,379 | Avg. Executed Instructions Per Scheduler [inst]    35.56
Issued Instructions [inst]                 11,384 | Avg. Issued Instructions Per Scheduler [inst]      35.58
```

Launch Statistics

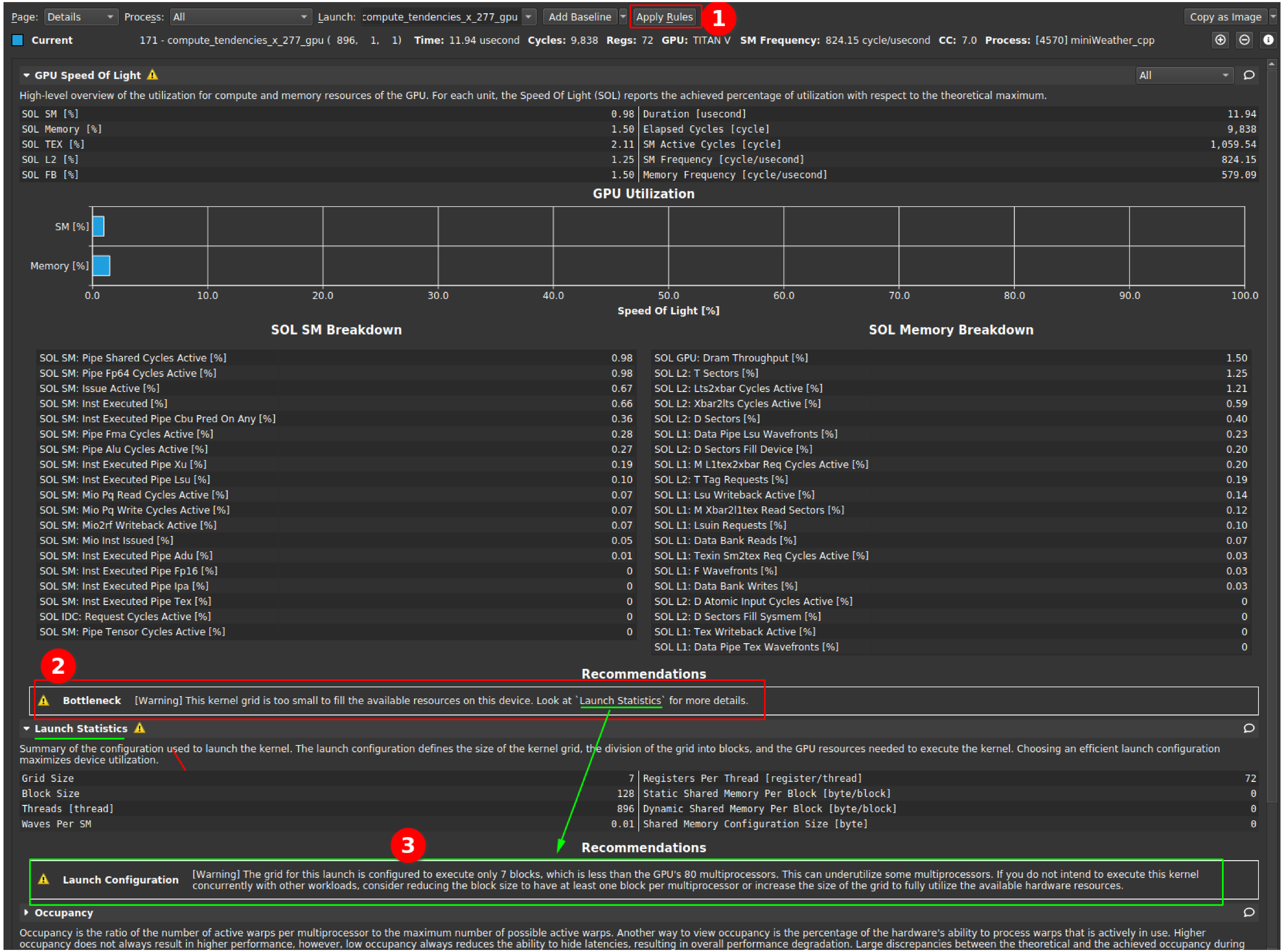
Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.

```
Grid Size                                       1 | Registers Per Thread [register/thread]                48
Block Size                                      4 | Static Shared Memory Per Block [byte/block]            0
Threads [thread]                                4 | Dynamic Shared Memory Per Block [byte/block]           0
Waves Per SM                                 0.00 | Shared Memory Configuration Size [byte]                0
```

Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

```
Theoretical Occupancy [%]                      50 | Block Limit Registers [block]                         40
Theoretical Active Warps per SM [warp/cycle]   32 | Block Limit Shared Mem [block]                        32
Achieved Occupancy [%]                       1.56 | Block Limit Warps [block]                             64
Achieved Active Warps Per SM [warp]          1.00 | Block Limit SM [block]                                32
```



Current 171 - compute_tendencies_x_277_gpu (896, 1, 1) **Time:** 11.94 usecond **Cycles:** 9,838 **Regs:** 72 **GPU:** TITAN V **SM Frequency:** 824.15 cycle/usecond **CC:** 7.0 **Process:** [4570] miniWeather_cpp

GPU Speed Of Light ⚠

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum.

SOL SM [%]	0.98	Duration [usecond]	11.94
SOL Memory [%]	1.50	Elapsed Cycles [cycle]	9,838
SOL TEX [%]	2.11	SM Active Cycles [cycle]	1,059.54
SOL L2 [%]	1.25	SM Frequency [cycle/usecond]	824.15
SOL FB [%]	1.50	Memory Frequency [cycle/usecond]	579.09

GPU Utilization

SOL SM Breakdown

SOL SM: Pipe Shared Cycles Active [%]	0.98
SOL SM: Pipe Fp64 Cycles Active [%]	0.98
SOL SM: Issue Active [%]	0.67
SOL SM: Inst Executed [%]	0.66
SOL SM: Inst Executed Pipe Cbu Pred On Any [%]	0.36
SOL SM: Pipe Fma Cycles Active [%]	0.28
SOL SM: Pipe Alu Cycles Active [%]	0.27
SOL SM: Inst Executed Pipe Xu [%]	0.19
SOL SM: Inst Executed Pipe Lsu [%]	0.10
SOL SM: Mio Pq Read Cycles Active [%]	0.07
SOL SM: Mio Pq Write Cycles Active [%]	0.07
SOL SM: Mio2rf Writeback Active [%]	0.07
SOL SM: Mio Inst Issued [%]	0.05
SOL SM: Inst Executed Pipe Adu [%]	0.01
SOL SM: Inst Executed Pipe Fp16 [%]	0
SOL SM: Inst Executed Pipe Ipa [%]	0
SOL SM: Inst Executed Pipe Tex [%]	0
SOL IDC: Request Cycles Active [%]	0
SOL SM: Pipe Tensor Cycles Active [%]	0

SOL Memory Breakdown

SOL GPU: Dram Throughput [%]	1.50
SOL L2: T Sectors [%]	1.25
SOL L2: Lts2xbar Cycles Active [%]	1.21
SOL L2: Xbar2lts Cycles Active [%]	0.59
SOL L2: D Sectors [%]	0.40
SOL L1: Data Pipe Lsu Wavefronts [%]	0.23
SOL L2: D Sectors Fill Device [%]	0.20
SOL L1: M L1tex2xbar Req Cycles Active [%]	0.20
SOL L2: T Tag Requests [%]	0.19
SOL L1: Lsu Writeback Active [%]	0.14
SOL L1: M Xbar2l1tex Read Sectors [%]	0.12
SOL L1: Lsuin Requests [%]	0.10
SOL L1: Data Bank Reads [%]	0.07
SOL L1: Texin Sm2tex Req Cycles Active [%]	0.03
SOL L1: F Wavefronts [%]	0.03
SOL L1: Data Bank Writes [%]	0.03
SOL L2: D Atomic Input Cycles Active [%]	0
SOL L2: D Sectors Fill Sysmem [%]	0
SOL L1: Tex Writeback Active [%]	0
SOL L1: Data Pipe Tex Wavefronts [%]	0

Recommendations

⚠ **Bottleneck** [Warning] This kernel grid is too small to fill the available resources on this device. Look at `Launch Statistics` for more details.

Launch Statistics ⚠

Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.

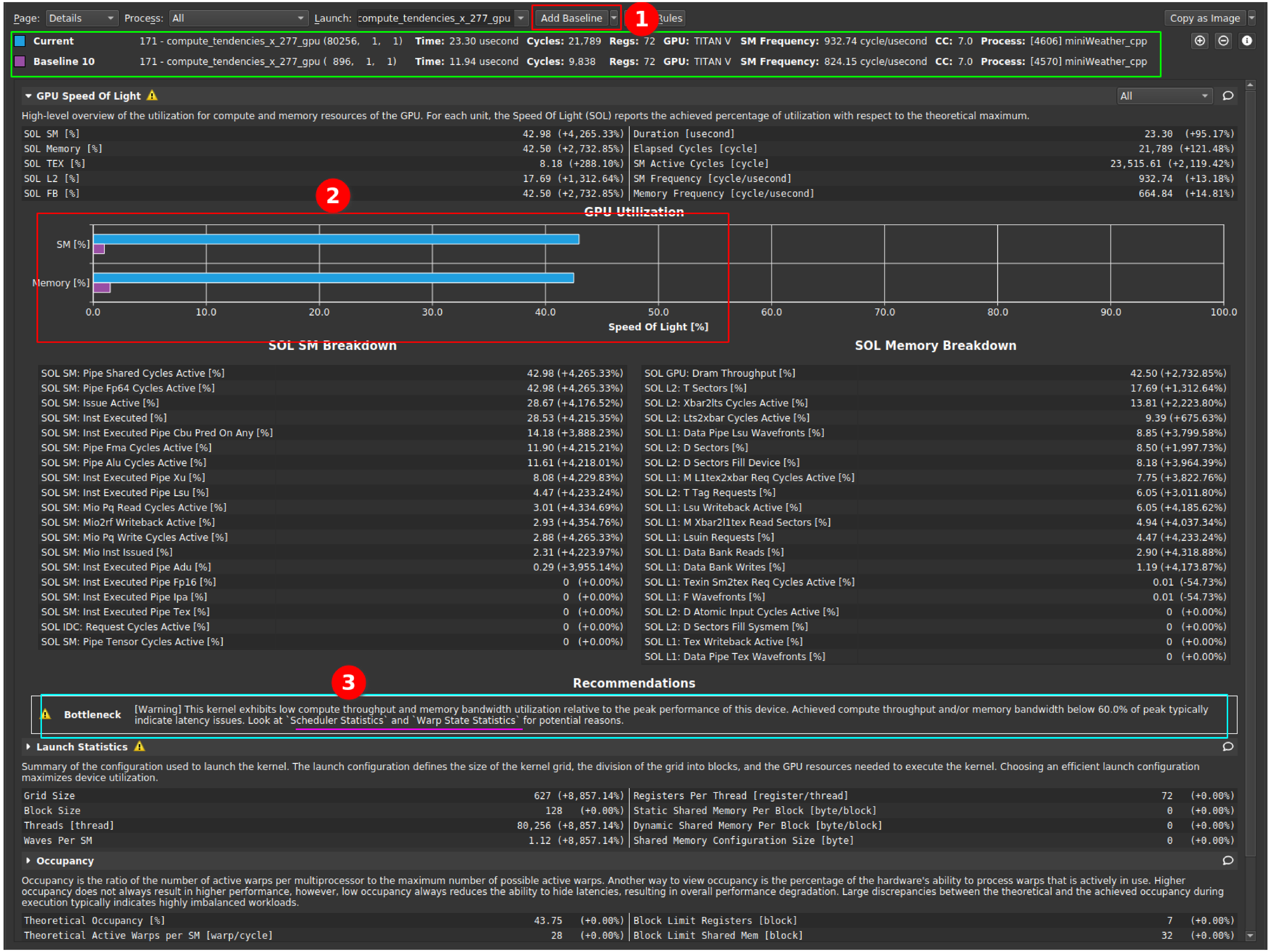
Grid Size	7	Registers Per Thread [register/thread]	72
Block Size	128	Static Shared Memory Per Block [byte/block]	0
Threads [thread]	896	Dynamic Shared Memory Per Block [byte/block]	0
Waves Per SM	0.01	Shared Memory Configuration Size [byte]	0

Recommendations

⚠ **Launch Configuration** [Warning] The grid for this launch is configured to execute only 7 blocks, which is less than the GPU's 80 multiprocessors. This can underutilize some multiprocessors. If you do not intend to execute this kernel concurrently with other workloads, consider reducing the block size to have at least one block per multiprocessor or increase the size of the grid to fully utilize the available hardware resources.

Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during



🟦 **Current**	171 - compute_tendencies_x_277_gpu (80256, 1, 1)	**Time:** 23.30 usecond	**Cycles:** 21,789	**Regs:** 72	**GPU:** TITAN V	**SM Frequency:** 932.74 cycle/usecond	**CC:** 7.0	**Process:** [4606] miniWeather_cpp
🟪 **Baseline 10**	171 - compute_tendencies_x_277_gpu (896, 1, 1)	**Time:** 11.94 usecond	**Cycles:** 9,838	**Regs:** 72	**GPU:** TITAN V	**SM Frequency:** 824.15 cycle/usecond	**CC:** 7.0	**Process:** [4570] miniWeather_cpp

GPU Speed Of Light ⚠

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum.

SOL SM [%]	42.98 (+4,265.33%)	Duration [usecond]	23.30 (+95.17%)
SOL Memory [%]	42.50 (+2,732.85%)	Elapsed Cycles [cycle]	21,789 (+121.48%)
SOL TEX [%]	8.18 (+288.10%)	SM Active Cycles [cycle]	23,515.61 (+2,119.42%)
SOL L2 [%]	17.69 (+1,312.64%)	SM Frequency [cycle/usecond]	932.74 (+13.18%)
SOL FB [%]	42.50 (+2,732.85%)	Memory Frequency [cycle/usecond]	664.84 (+14.81%)

SOL SM Breakdown

SOL SM: Pipe Shared Cycles Active [%]	42.98 (+4,265.33%)
SOL SM: Pipe Fp64 Cycles Active [%]	42.98 (+4,265.33%)
SOL SM: Issue Active [%]	28.67 (+4,176.52%)
SOL SM: Inst Executed [%]	28.53 (+4,215.35%)
SOL SM: Inst Executed Pipe Cbu Pred On Any [%]	14.18 (+3,888.23%)
SOL SM: Pipe Fma Cycles Active [%]	11.90 (+4,215.21%)
SOL SM: Pipe Alu Cycles Active [%]	11.61 (+4,218.01%)
SOL SM: Inst Executed Pipe Xu [%]	8.08 (+4,229.83%)
SOL SM: Inst Executed Pipe Lsu [%]	4.47 (+4,233.24%)
SOL SM: Mio Pq Read Cycles Active [%]	3.01 (+4,334.69%)
SOL SM: Mio2rf Writeback Active [%]	2.93 (+4,354.76%)
SOL SM: Mio Pq Write Cycles Active [%]	2.88 (+4,265.33%)
SOL SM: Mio Inst Issued [%]	2.31 (+4,223.97%)
SOL SM: Inst Executed Pipe Adu [%]	0.29 (+3,955.14%)
SOL SM: Inst Executed Pipe Fp16 [%]	0 (+0.00%)
SOL SM: Inst Executed Pipe Ipa [%]	0 (+0.00%)
SOL SM: Inst Executed Pipe Tex [%]	0 (+0.00%)
SOL IDC: Request Cycles Active [%]	0 (+0.00%)
SOL SM: Pipe Tensor Cycles Active [%]	0 (+0.00%)

SOL Memory Breakdown

SOL GPU: Dram Throughput [%]	42.50 (+2,732.85%)
SOL L2: T Sectors [%]	17.69 (+1,312.64%)
SOL L2: Xbar2lts Cycles Active [%]	13.81 (+2,223.80%)
SOL L2: Lts2xbar Cycles Active [%]	9.39 (+675.63%)
SOL L1: Data Pipe Lsu Wavefronts [%]	8.85 (+3,799.58%)
SOL L2: D Sectors [%]	8.50 (+1,997.73%)
SOL L2: D Sectors Fill Device [%]	8.18 (+3,964.39%)
SOL L1: M L1tex2xbar Req Cycles Active [%]	7.75 (+3,822.76%)
SOL L2: T Tag Requests [%]	6.05 (+3,011.80%)
SOL L1: Lsu Writeback Active [%]	6.05 (+4,185.62%)
SOL L1: M Xbar2l1tex Read Sectors [%]	4.94 (+4,037.34%)
SOL L1: Lsuin Requests [%]	4.47 (+4,233.24%)
SOL L1: Data Bank Reads [%]	2.90 (+4,318.88%)
SOL L1: Data Bank Writes [%]	1.19 (+4,173.87%)
SOL L1: Texin Sm2tex Req Cycles Active [%]	0.01 (-54.73%)
SOL L1: F Wavefronts [%]	0.01 (-54.73%)
SOL L2: D Atomic Input Cycles Active [%]	0 (+0.00%)
SOL L2: D Sectors Fill Sysmem [%]	0 (+0.00%)
SOL L1: Tex Writeback Active [%]	0 (+0.00%)
SOL L1: Data Pipe Tex Wavefronts [%]	0 (+0.00%)

Recommendations

⚠ **Bottleneck** [Warning] This kernel exhibits low compute throughput and memory bandwidth utilization relative to the peak performance of this device. Achieved compute throughput and/or memory bandwidth below 60.0% of peak typically indicate latency issues. Look at `Scheduler Statistics` and `Warp State Statistics` for potential reasons.

Launch Statistics ⚠

Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.

Grid Size	627 (+8,857.14%)	Registers Per Thread [register/thread]	72 (+0.00%)
Block Size	128 (+0.00%)	Static Shared Memory Per Block [byte/block]	0 (+0.00%)
Threads [thread]	80,256 (+8,857.14%)	Dynamic Shared Memory Per Block [byte/block]	0 (+0.00%)
Waves Per SM	1.12 (+8,857.14%)	Shared Memory Configuration Size [byte]	0 (+0.00%)

Occupancy

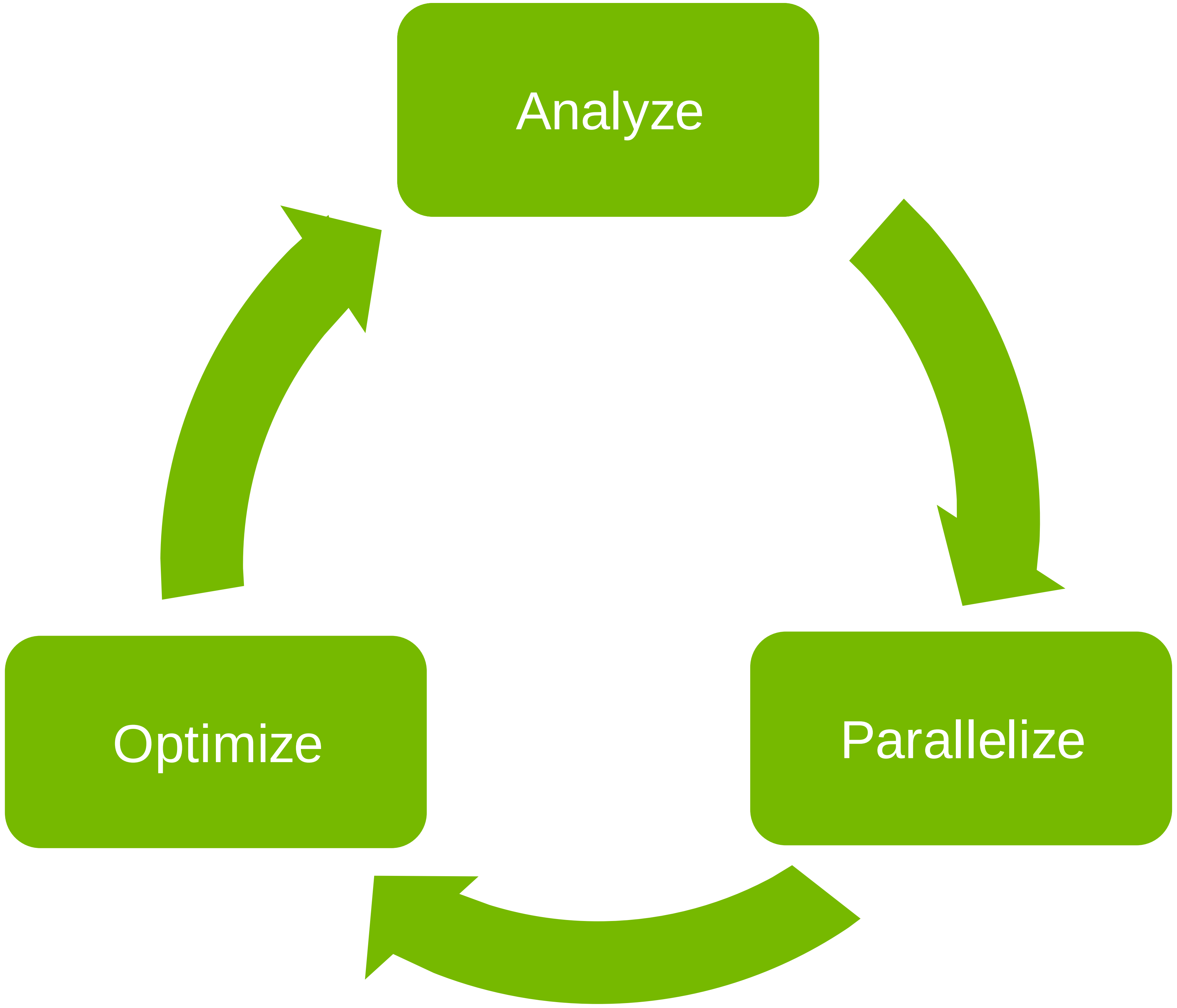
Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

Theoretical Occupancy [%]	43.75 (+0.00%)	Block Limit Registers [block]	7 (+0.00%)
Theoretical Active Warps per SM [warp/cycle]	28 (+0.00%)	Block Limit Shared Mem [block]	32 (+0.00%)

SUMMARY

Performance Optimization is a Constant Learning Process

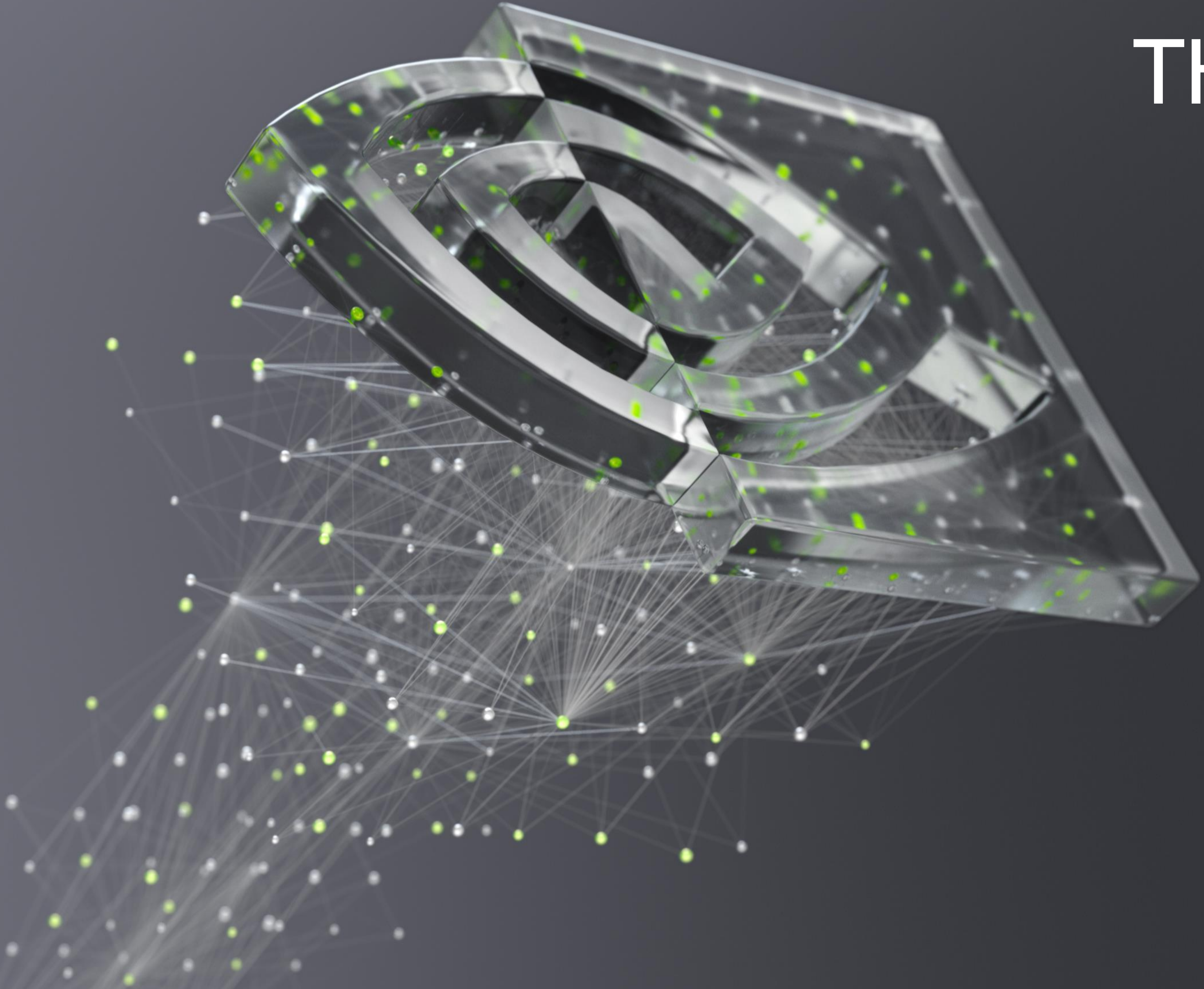
1. Know your application
2. Know your hardware
3. Know your tools
4. Know your process
    1. Identify the Hotspot
    2. Classify the Performance Limiter
    3. Look for indicators



ADDITIONAL REFERENCES

- Documentation
  - https://www.openacc.org/resources
  - https://docs.nvidia.com/cuda/cuda-c-programming-guide/
  - https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/
- GTC 2020 Sessions:
  - What the Profiler is Telling You: How to Get the Most Performance out of Your Hardware [S22141]
    - https://www.nvidia.com/en-us/gtc/on-demand/?search=s22141
- More:
  - NVIDIA Nsight Compute
    - https://vimeo.com/398929189
  - NVIDIA Nsight Systems
    - https://vimeo.com/398838139

THANK YOU!



NVIDIA