



**GPU PROFILING**



# AGENDA

## The need for GPU Profiling

- Why Profile Your Code
- When to Profile Your Code
- Are there opportunities to improve performance?

---

## GPU Profiling Tools

### Nsight Suite of Profilers

- Nsight Systems
  - NVTX: NVIDIA Tools Extension API
- Nsight Compute

---

## Profiling Best Practices: Tips and Techniques for Efficient Workloads

---

## Resources



# WHY PROFILE YOUR CODE?

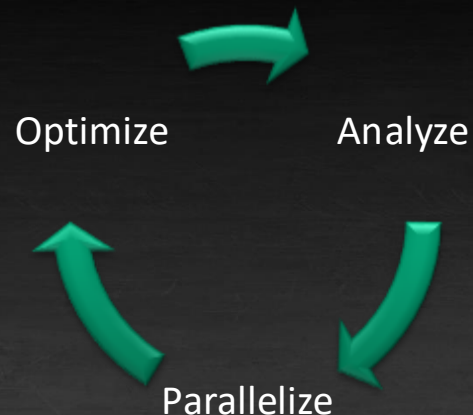
When is this useful?

- Code profiling gives you **deeper insights** into how your code performs at runtime.
- Allows you to focus your efforts on the **problem areas** of your application if optimizations are needed.
- Optimization does not always refer to execution time or latency, you can also **optimize memory usage or throughput**.
- Profiling can help you **find errors** or easy optimization opportunities.
- Often the way code executes on a device doesn't match what we would expect based on reading the code - profiling can help you figure out why.

# WHEN TO PROFILE YOUR CODE

Is it time to profile yet?

- Focus your efforts initially on getting a **readable, maintainable, and correct version** of your program finished.
  - **Correctness** beats everything else, nobody cares how fast or memory efficient your program is if it is unstable or gives the wrong result.
- **Avoid premature optimization**, bottlenecks can typically be attributed to a small fraction (<10%) of the overall code. Humans often aren't great at predicting which parts of the code that will be.
- Make sure your code is **modular**, break tasks into small easily readable functions with descriptive names.
- Decide what to optimize, it may not be latency/execution time. It might be more important to optimize for memory usage or cost/throughput.
- Follow the Development Cycle:
  - Analyze
  - Parallelize
  - Optimize



# FINDING PERFORMANCE OPPORTUNITIES

Models can be data bound by the data pipeline, compute or memory

- GPU utilization as it relates to model code
  - Time being spent on ops in every iteration
  - Time spent on GPU/CPU
  - Data types used for operations
- Bottlenecks could be attributed to
  - Input data pipeline: data loading, preprocessing etc
  - Compute (math) limited operations
  - Memory limited operations
  - Other aspects such as overall system tuning
- Categories of operations in DNNs based on bottleneck
  - Element wise: ReLU, memory bound
  - Reduction: Batch norm, memory bound
  - Dot product: Convolution, math bound

Compute bound



Memory bound



Compute heavy ops see speed-ups from GPUs

# DEEP LEARNING OPTIMIZATION

## Performance Analysis at System and DNN Level & Visualization

### System Level Tuning

- System Tuning
  - Thread Synchronization, Multi GPU and node communication
  - Memory management & Kernel profiling
- Leveraging/Optimizing Hardware
- Input Pipeline Optimization
- Many others....

### DNN Level Tuning

- Algorithm Techniques & Data Representations
- Pruning
- Calibration
- Quantization
- Many others....

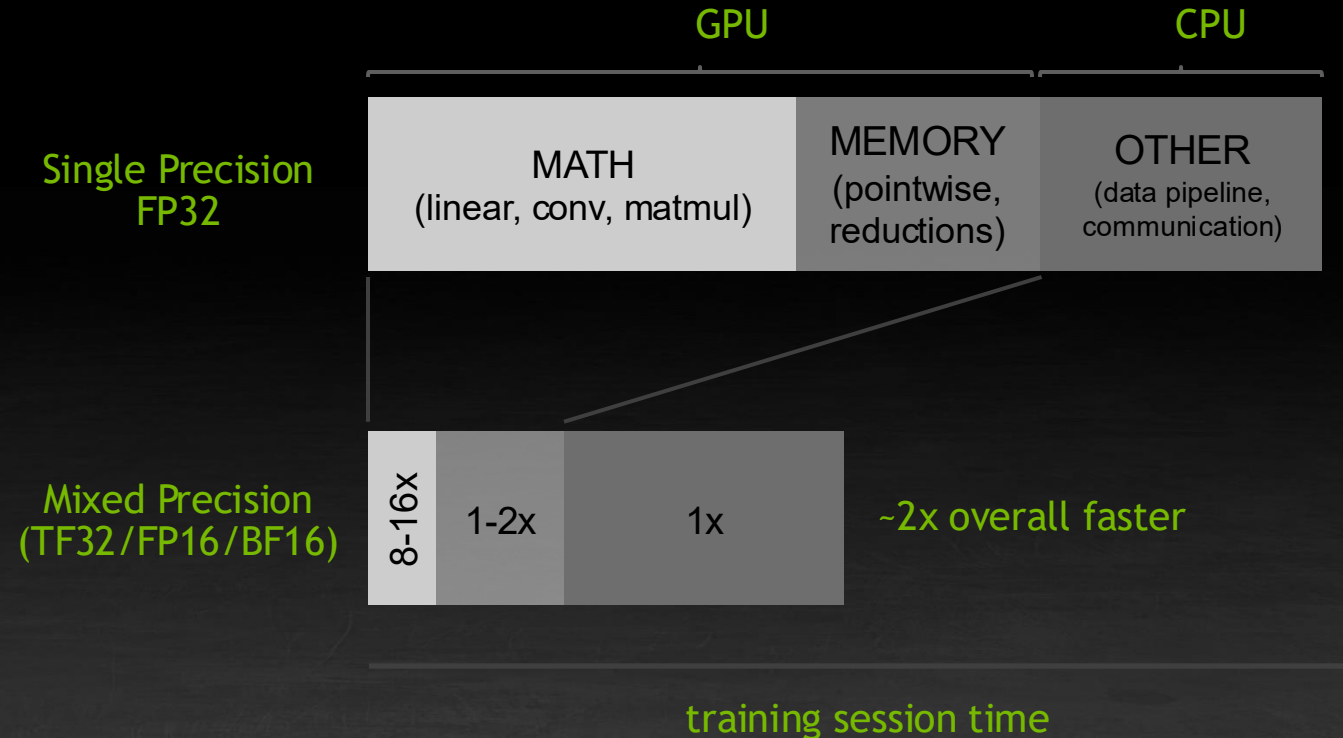


# LIMITS OF PERFORMANCE OPTIMIZATION

End-to-end perf depends on training composition

- Amdahl's Law:

If you speed up part of your training session (GPU work), then the remaining parts (CPU work) limit your overall performance



# DL PROFILING NEEDS OF DIFFERENT PERSONAS

## Researchers



Fast development of best performant models for research, challenge and domains

## Data Scientists & Applied Researchers



Reduce Training time, focus on data, develop and apply the best models for the applications

## Sysadmins & DevOps



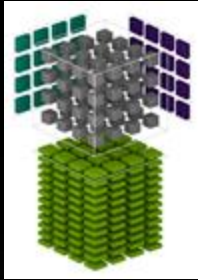
Optimized utilization and uptime, monitor GPU workloads, leverage hardware



# TOOLS & TECHNOLOGIES

Profiling and monitoring tools for all users

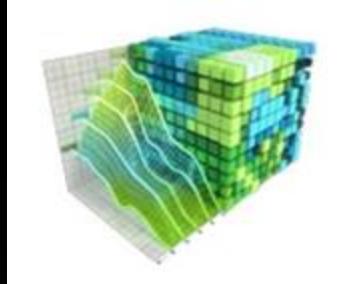
Researchers &  
Developers



NVTX  
Nsight Systems  
Nsight Compute

Skills in Algorithms

Data Scientists & Applied  
Researchers



Nsight Systems with NVTX

Skills in Domains & Applications

Sysadmins &  
DevOps



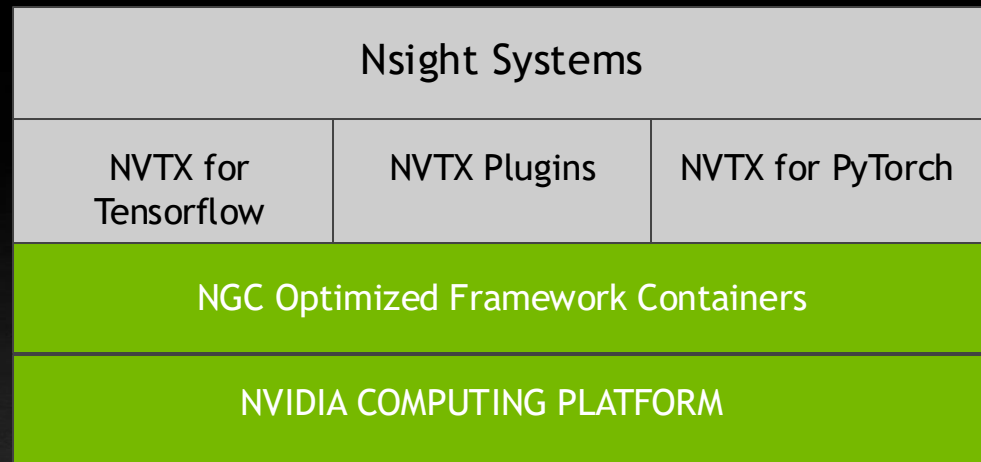
Data Center Monitoring Tools  
**DCGM**, NVML

Skills in Systems

# NVIDIA PROFILING STACK

The layers that make the cake

- **Nsight Systems** and **Nsight Compute** have been built using CUDA Profiling Tools Interface(CUPTI)
- **NVTX** NVIDIA Tools Extension Library is a way to annotate source code with markers
  - NVTX markers are used to annotate and focus on sections of code important to the user
  - TensorFlow optimized by NVIDIA (aka nvidia-tensorflow) contain support for NVTX markers.
- NVTX plugins are python bindings for users to add markers easily



A close-up photograph of a green microchip mounted on a dark, textured substrate. The chip features a grid of numerous small, square, green components. The lighting is dramatic, with a strong light source from the upper right, creating a bright, circular bokeh effect in the background and highlighting the edges of the chip's components. The foreground shows the sharp, vertical edges of the chip's pins or components.

**NSIGHT SUITE OF PROFILERS**



# NSIGHT PRODUCT FAMILY

## Standalone Performance Tools, IDE Plugins

### Standalone Performance Tools

**Nsight Systems** System wide tracing, application algorithm tuning

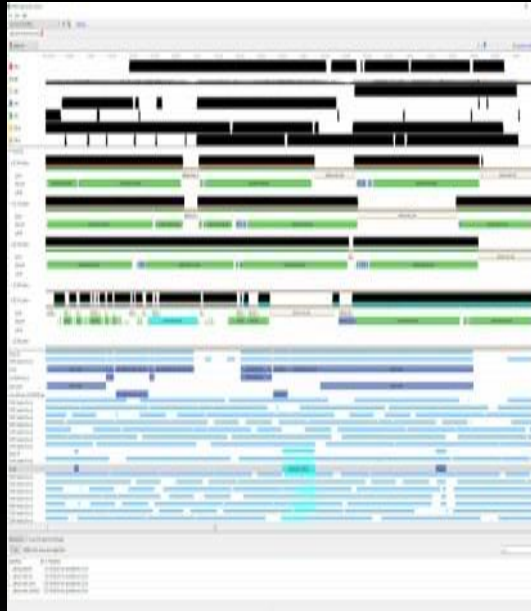
**Nsight Compute** Debug/Optimize specific CUDA kernels

**Nsight Graphics** Debug/Optimize specific graphics API and Shaders

### IDE Plugins

**Nsight Visual Studio/Eclipse Edition** editor, debugger, performance analysis

# NSIGHT PRODUCT FAMILY



## Nsight Systems

System-wide application  
algorithm tuning



## Nsight Compute

CUDA API Debugging & Kernel  
Profiling



## Nsight Graphics

Graphics Debugging & Profiling

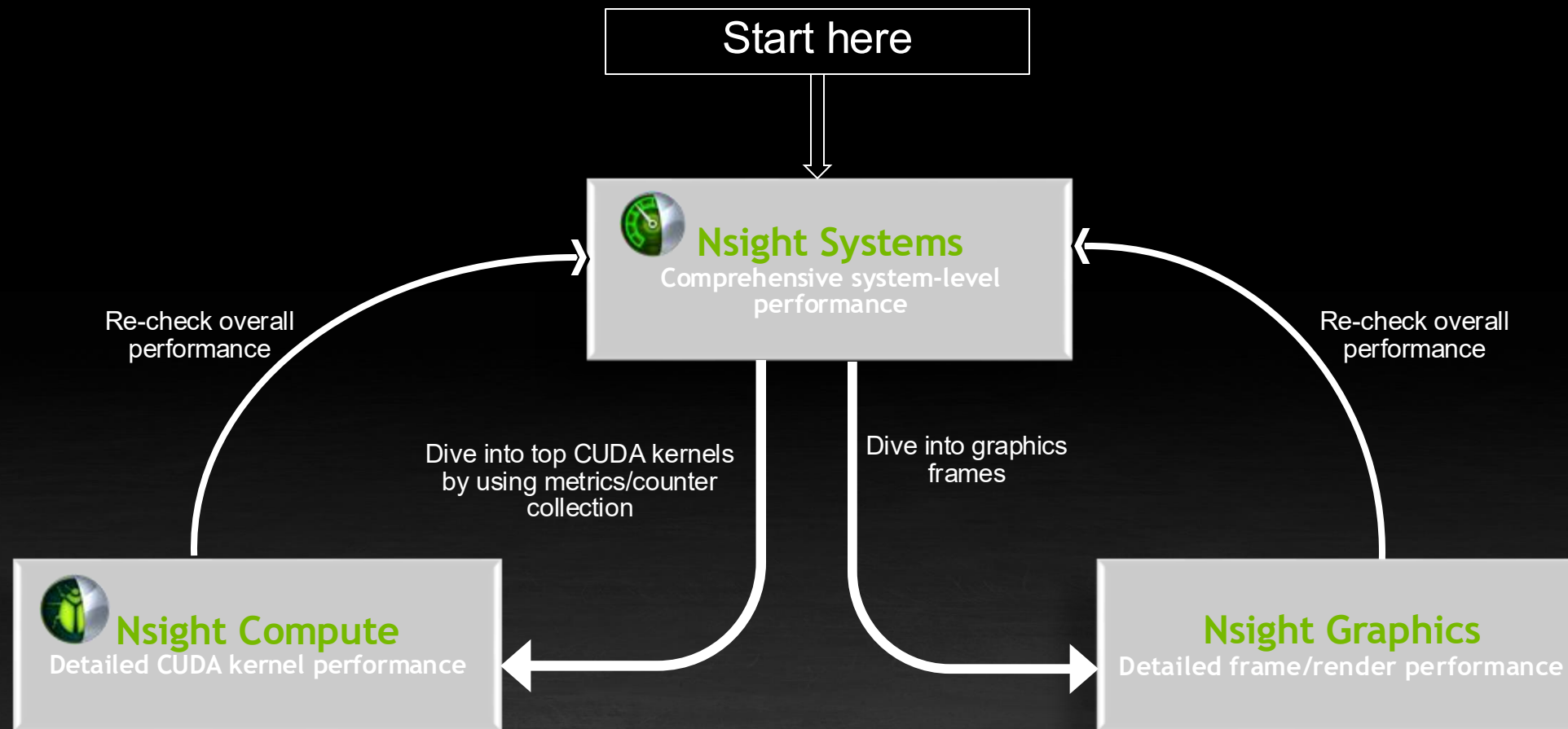


## IDE Plugins

Nsight Eclipse  
Edition/Visual Studio  
(Editor, Debugger)

# NSIGHT (STANDALONE) TOOLS WORKFLOW

Nsight Systems, Nsight Compute and Nsight Graphics





# NSIGHT SYSTEMS

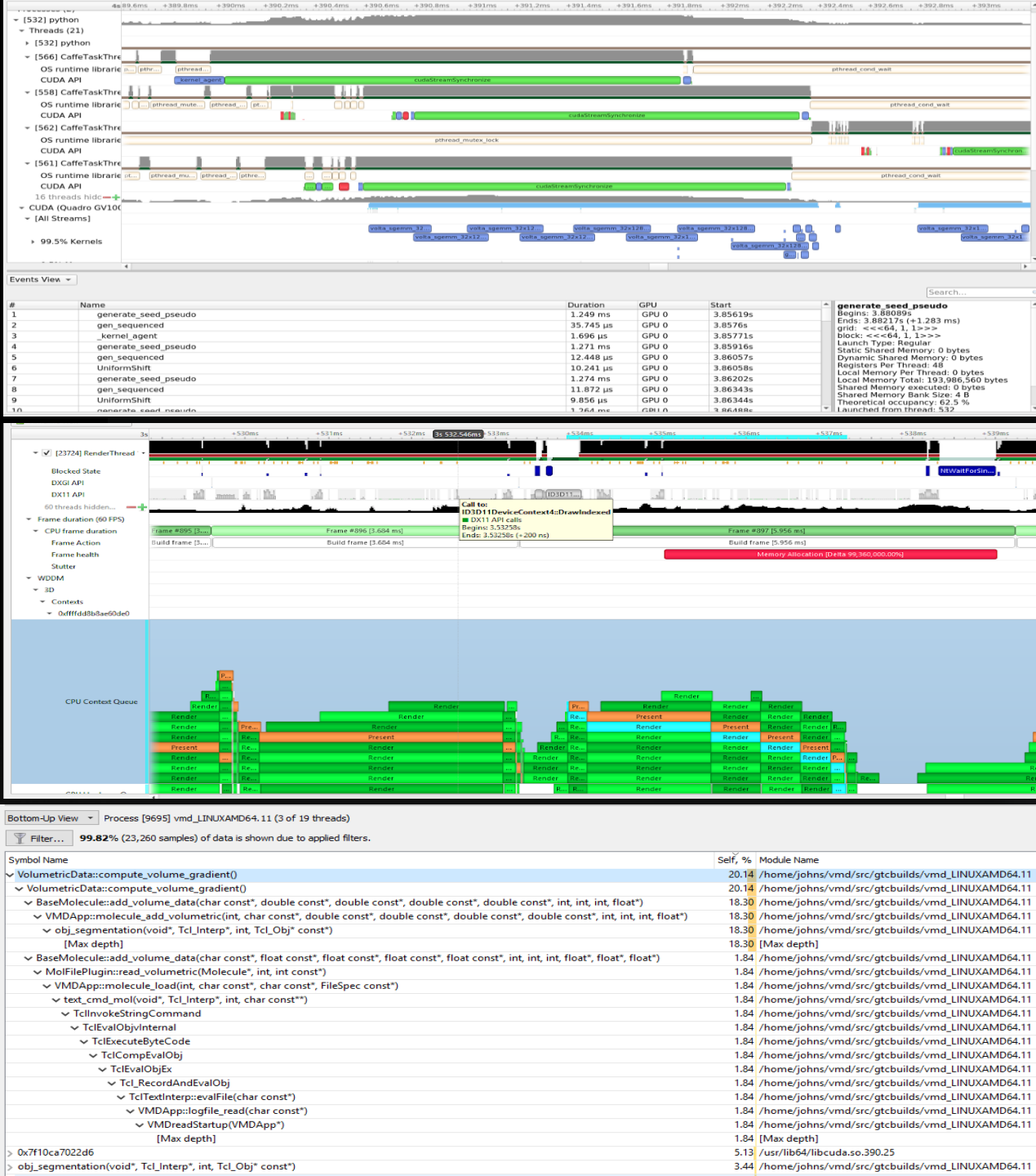
## System Profiler

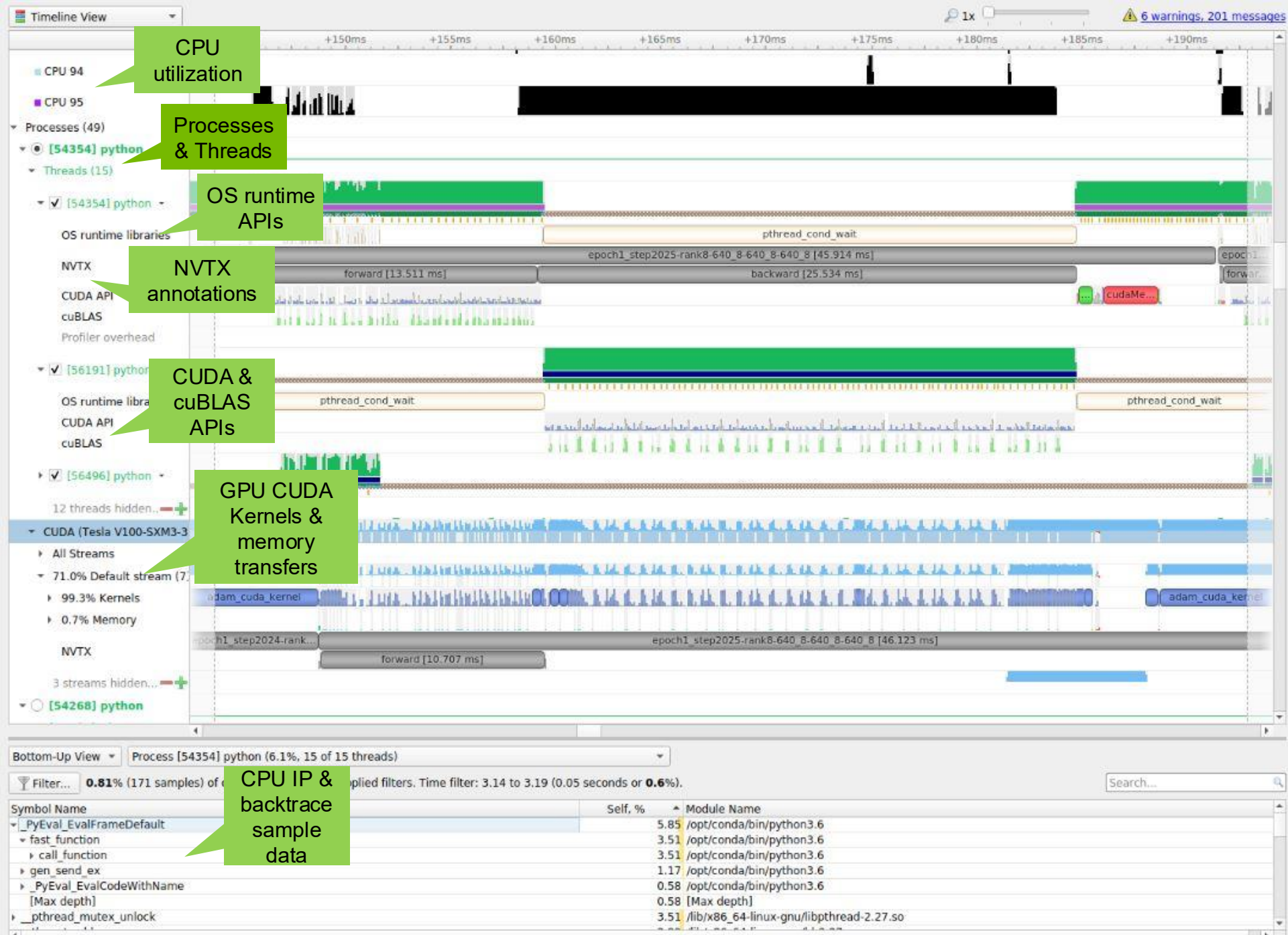
### Key Features:

- 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)

Documentation: <https://developer.nvidia.com/nsight-systems>





## Additionally:

### Trace:

- TensorRT
- Direct3D11,12,DXR
- Vulkan
- OpenGL
- OpenACC
- MPI
- OpenMP
- Ftrace
- ETW
- WDDM
- GPU Context Switch

### Export:

- SQLite
- HDF5
- JSON

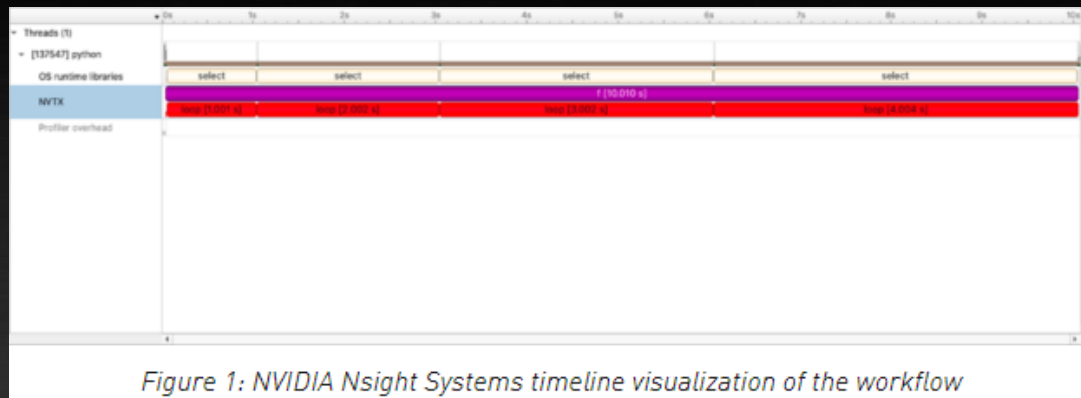
### Architectures:

- X86\_64
- Power
- Arm SBSA
- Tegra

# NVTX: NVIDIA TOOLS EXTENSION API

An Annotation Tool for Profiling Code in Python and C/C++

- Code Annotation tool to mark functions or chunks of code.
- NVTX and Nsight Systems together are powerful tools for visualizing CPU and GPU performance
- Workflow:
  - Use decorators `@nvtx.annotate()` or context manager `with nvtx.annotate(..):` to mark code to be measured.
  - Run it with Nsight Systems.
    - `$ nsys profile -t nvtx,osrt --force-overwrite=true --stats=true --output=quickstart python nvtx-quickstart.py`
    - `$ nsys -help` or `nsys [specific command] --help`
  - Qdrep file and sqlite database generated to be viewed on Nsight Systems UI.



```
import time
import nvtx

@nvtx.annotate("f()", color="purple")
def f():
    for i in range(5):
        with nvtx.annotate("loop", color="red"):
            time.sleep(i)

f()
```



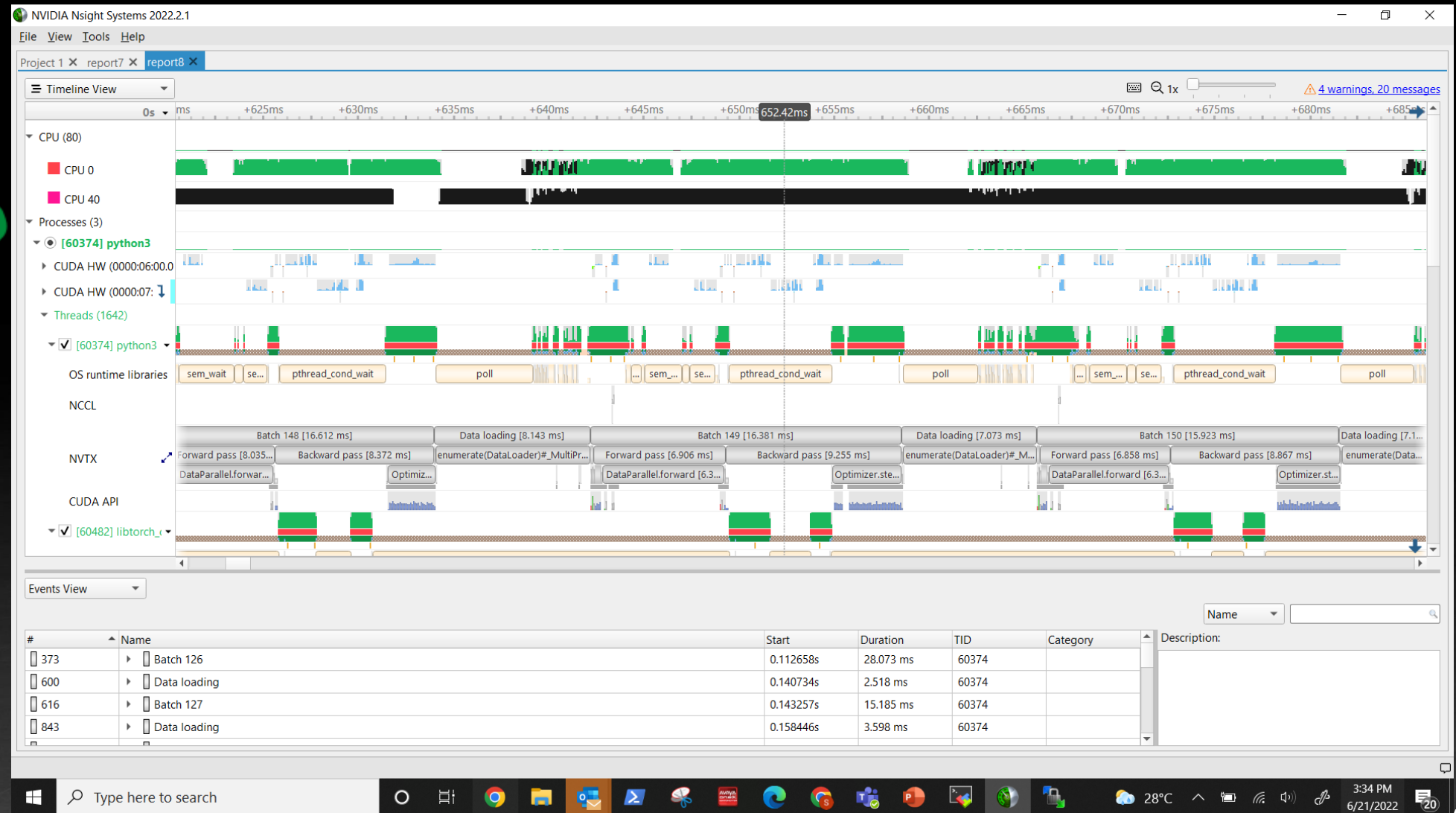
# NSIGHT SYSTEMS

## Key Features

■ Timeline View



■ Event table



## Analysis Summary

The screenshot displays the NVIDIA Nsight Systems 2022.2.1 application window. The top menu bar includes 'File', 'View', 'Tools', and 'Help'. Below the menu, there are tabs for 'Project 1', 'report7', and 'report8', with 'report8' being the active tab. A dropdown menu labeled 'Analysis Summary' is visible. The main content area shows the 'Profiling session duration: 00:20.000' and a table of session details. Below this, a section for 'dgx1-003 (0:0)' contains a 'Target' table with system and hardware information. The Windows taskbar at the bottom shows the search bar and various application icons. The system tray in the bottom right corner displays the time as 3:38 PM on 6/21/2022 and the temperature as 28°C.

## Diagnosics Summary

NVIDIA Nsight Systems 2022.2.1

File

View

Tools

Help

Project 1

report7

report8

Diagnostics Summary

Messages

	Source	Process ID	Time	Description
i	Daemon	60374	-00:30.702	Process was launched by the profiler, see <a href="#">/tmp/nvidia/nsight_systems/quadd_session_1060356/streams/pid_60374_stdout.log</a> and <a href="#">stderr.log</a> for program output
i	Injection	60374	-00:30.612	Common injection library initialized successfully.
i	Injection	60374	-00:30.597	OS runtime libraries injection initialized successfully.
i	Injection	60374	-00:30.358	cuDNN injection initialized successfully.
i	Injection	60374	-00:30.159	OpenMP injection initialized successfully.
i	Injection	60374	-00:29.719	Buffers holding CUDA trace data will be flushed on CudaProfilerStop() call.
i	Injection	60374	-00:29.451	CUDA injection initialized successfully.
i	Injection	60374	-00:28.117	NVTX injection initialized successfully.
i	Daemon		-00:00.188	Intel(c) Last Branch Record (LBR) backtraces collected.
i	Daemon		-00:00.188	Hardware event 'instructions', with sampling period 1800000, used to trigger sample collection.
i	Daemon		-00:00.000	1 CPU IP samples collected for every CPU IP backtrace collected.
i	Analysis		00:00.000	Profiling has started.
w	Analysis	60374	00:04.280	Not all NVTX events might have been collected.
i	Analysis	60374	00:04.280	Number of NVTX events collected: 434,574.
w	Analysis	60374	00:04.280	Not all CUDA events might have been collected.
i	Analysis	60374	00:04.280	Number of CUDA events collected: 374,419.
w	Analysis	60374	00:04.280	Not all OS runtime libraries events might have been collected.
i	Analysis	60374	00:04.280	Number of OS runtime libraries events collected: 50,873.
w	Analysis	60374	00:04.280	Not all cuDNN events might have been collected.
i	Analysis	60374	00:04.280	Number of cuDNN events collected: 8,140.
i	Analysis		00:20.000	Profiling has stopped.
i	Injection	60374	00:20.067	Number of CUPTI events produced: 397,345, CUPTI buffers: 50.
i	Injection	60374	00:20.067	The cudaProfilerStart API was ignored 1 times due to configuration settings; Note that, when requested, only the first pair of cudaProfilerStart/Stop APIs, after the collection is started, will be effective.
i	Daemon		00:20.901	Number of IP samples collected: 30,878.

Windows Taskbar

Type here to search

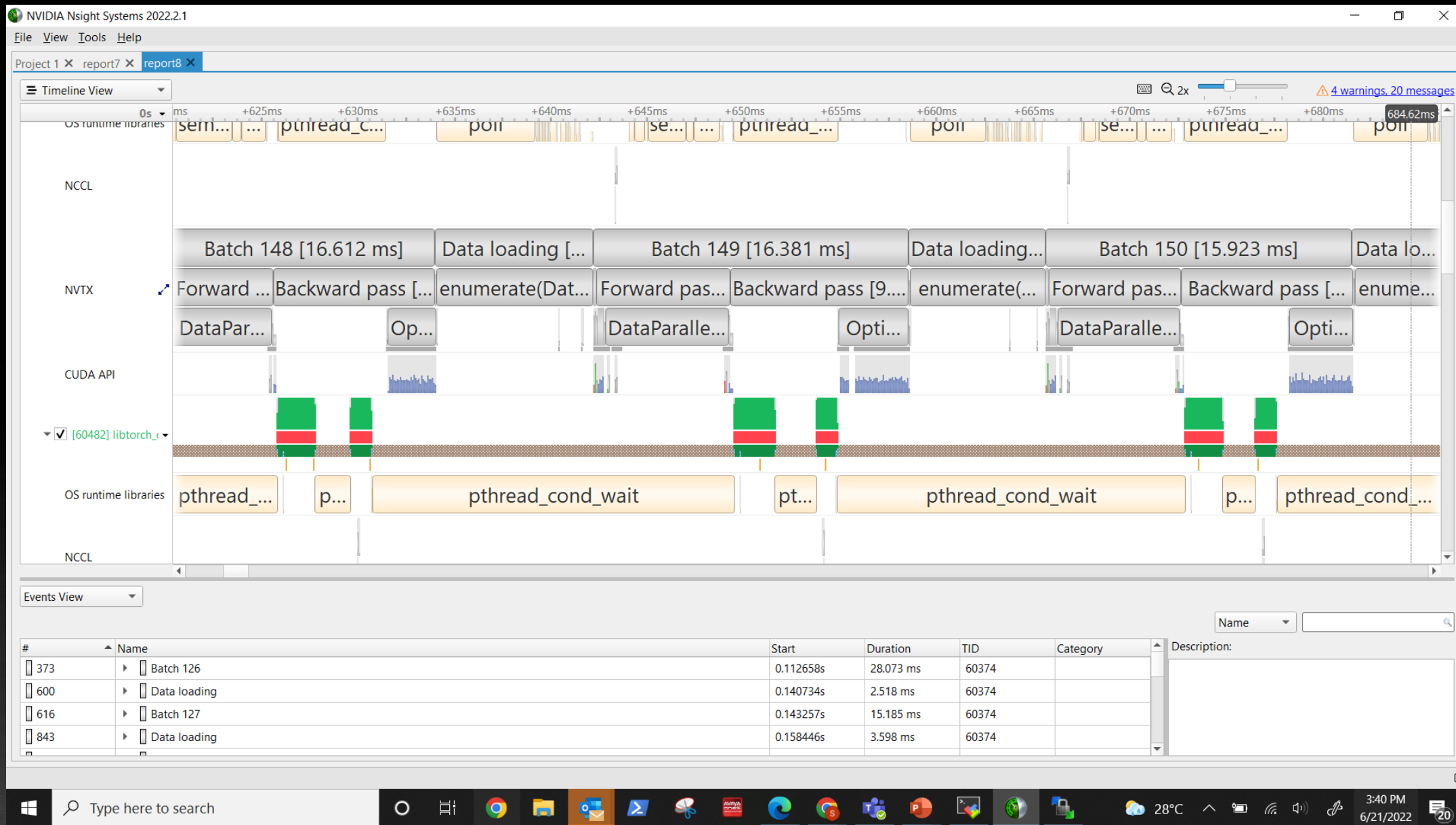
Taskbar Icons

System Tray

3:38 PM 6/21/2022



# NVTX Trace



## Other Features

### CLI improvement

Simultaneous sessions & management

`nsys launch` forwards terminal and signals

`nsys stats` command

`nsys export` command

CUDA context “All Streams” aggregation tree

Windows WDDM GPU memory usage, paging, evictions

Graphics trace enhancements

```
1 2 3 4 5
Terminal
16:23:58 $
16:24:07 $ nsys profile --output /tmp/smoke --show-output false ./smokeParticles
Collecting data...
█

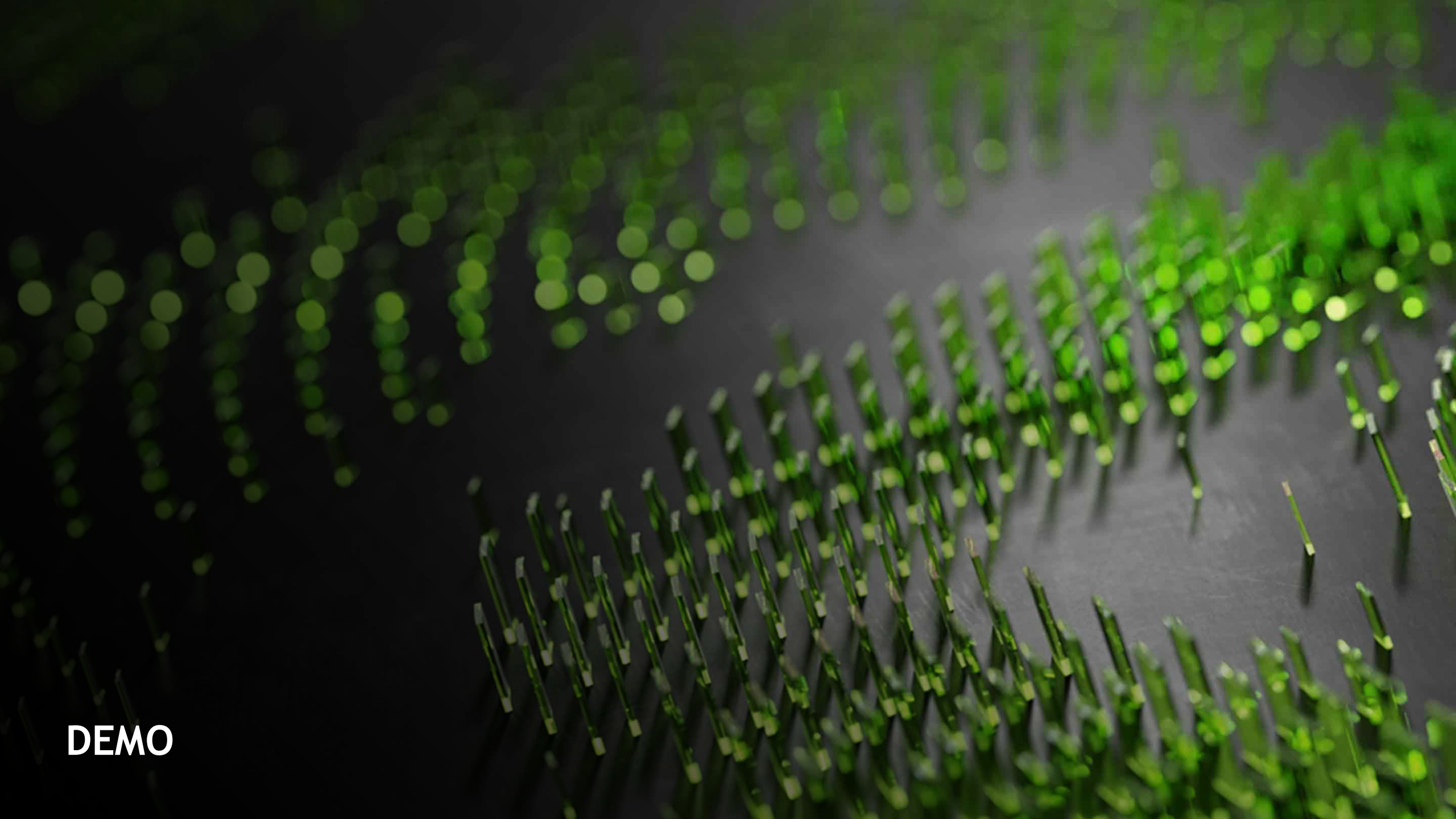
Terminal
16:24:18 $ nsys start --session-new particles --output /tmp/particles
start executed. use the 'launch' command to start the collection
16:24:22 $ nsys launch --session particles --show-output false ./particles
application launched

The target application terminated with signal 15 (SIGTERM)
█

Terminal
16:23:51 $
16:24:28 $ nsys launch --trace cuda --sample none --session-new fog --show-output false ./randomFog
WARNING: Backtraces will not be collected because sampling is disabled.
application launched
█

Terminal
ID          TIME          STATE LAUNCH NAME
1016352     00:25          Collecting 1 profile-smokeParticles-16352
1116423     00:15          Collecting 1 particles
1216488     00:04          WaitForStart 1 fog
16:24:40 $ nsys cancel --session 1016352
cancel executed
16:24:42 $ nsys sessions list
ID          TIME          STATE LAUNCH NAME
1016352     00:31          WaitForStart 1 profile-smokeParticles-16352
1116423     00:21          Collecting 1 particles
1216488     00:10          WaitForStart 1 fog
16:24:46 $ nsys start --output /tmp/fog --session fog
start executed
16:24:50 $ nsys stop --session fog
Processing events...
Saving intermediate "/tmp/fog.qdstrm" file to disk...

Importing [=====100%]
Saved report file to "/tmp/fog.qdrep"
stop executed
16:24:54 $ nsys sessions list
ID          TIME          STATE LAUNCH NAME
1016352     00:43          WaitForStart 1 profile-smokeParticles-16352
1116423     00:33          Collecting 1 particles
1216488     00:22          WaitForStart 1 fog
16:24:57 $ nsys shutdown --session particles
collection cancelled. shutdown executed
16:25:01 $ nsys sessions list
ID          TIME          STATE LAUNCH NAME
1016352     00:50          WaitForStart 1 profile-smokeParticles-16352
1216488     00:29          WaitForStart 1 fog
█
```



DEMO



# NSIGHT COMPUTE

## Kernel Profiling Tool

### Key Features:

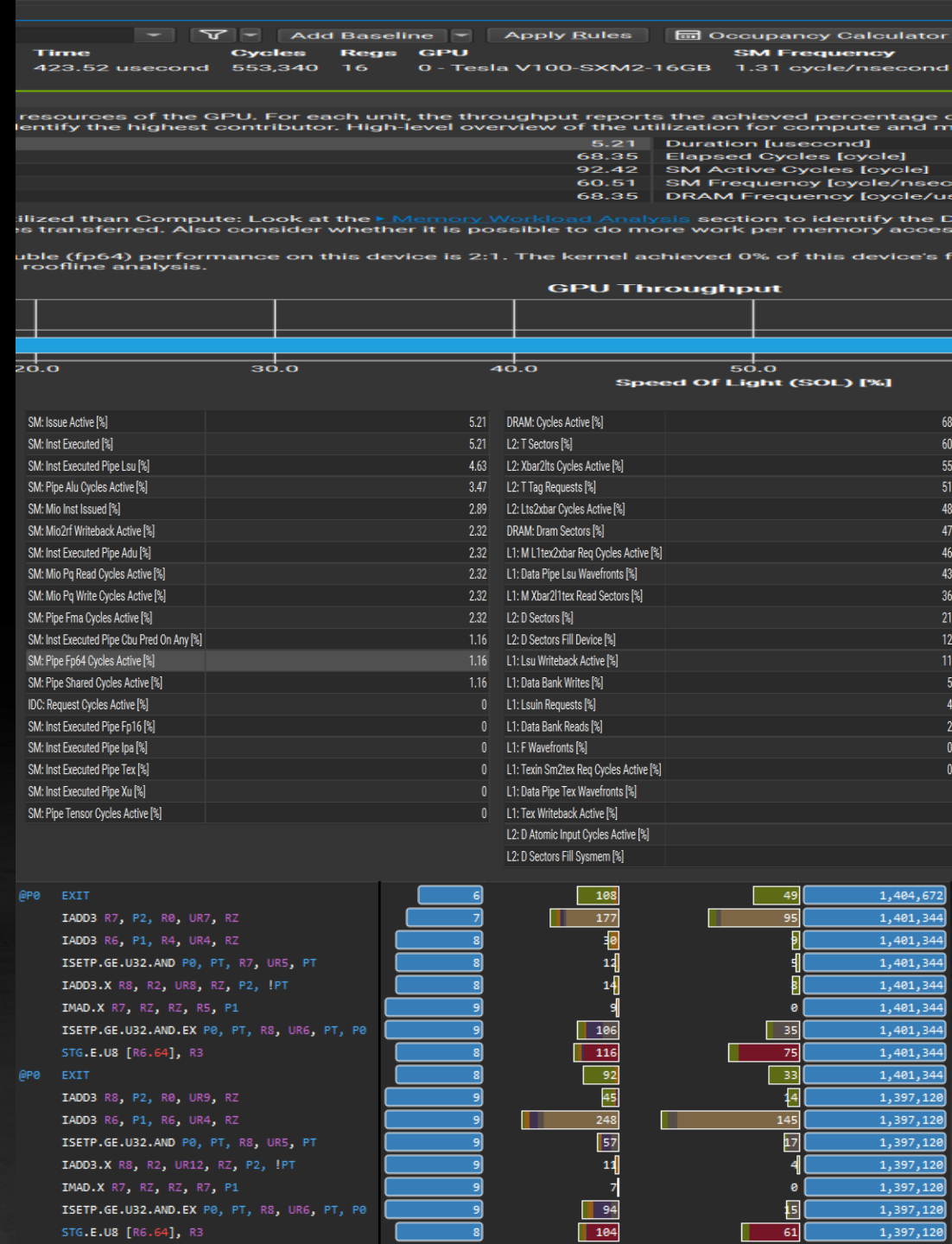
- Interactive CUDA API debugging and kernel profiling
- Fast Data Collection
- Compare performance metrics across different runs

```
$ ncu --set full -k <kernel_name> -s 4 -c 1 -o myreport ./myapp
```

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

GPUs: Volta, Turing, A100 GPUs

Documentation: <https://developer.nvidia.com/nsight-compute>

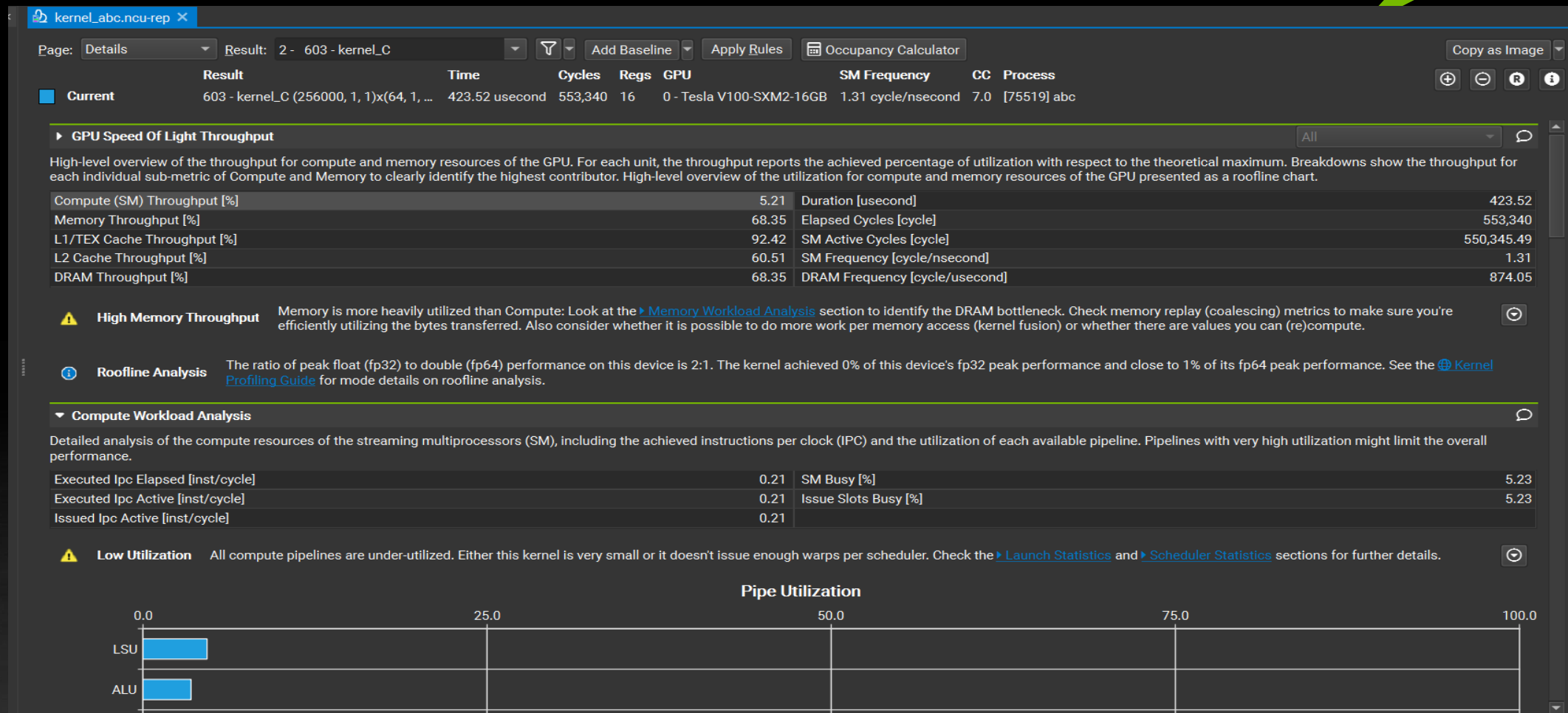




# NSIGHT COMPUTE

## Profile Report - Details Page

All Data on  
Single Page



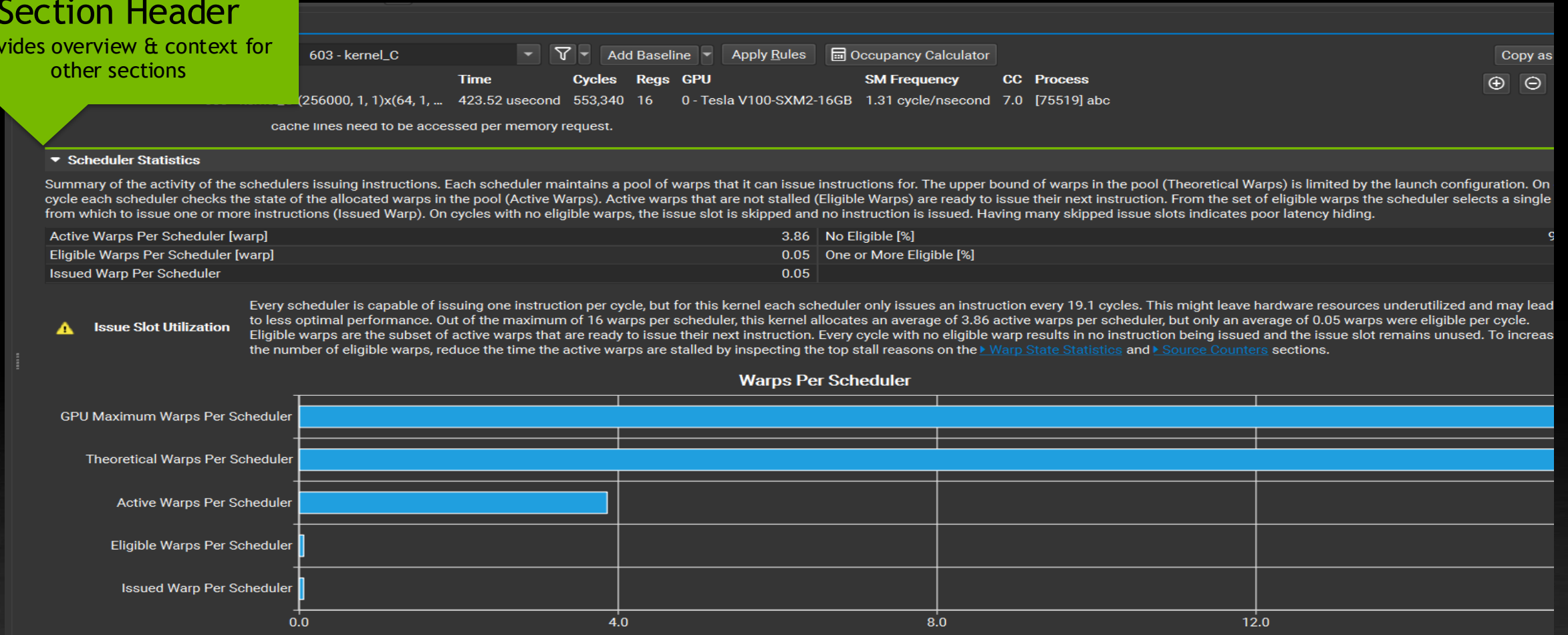
Ordered from Top-Level to Low-Level

# NSIGHT COMPUTE

## Section Example

### Section Header

provides overview & context for other sections



### Section Config

completely data driven  
add/modify/change sections

# NSIGHT COMPUTE

## Unguided Analysis / Rules System

kernel\_abc.ncu-rep

Page: Details

Result: 2 - 603 - kernel\_C



Add Baseline

Apply Rules

Occupancy Calculator

	Result	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	603 - kernel_C (256000, 1, 1)x(64, 1, ...	423.52 usecond	553,340	16	0 - Tesla V100-SXM2-16GB	1.31 cycle/nsecond	7.0	[75519] abc

### GPU Speed Of Light Throughput

All

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

Compute (SM) Throughput [%]	5.21	Duration [usecond]
	68.35	Elapsed Cycles [cycle]
	92.42	SM Active Cycles [cycle]
	60.51	SM Frequency [cycle/nsecond]
	68.35	DRAM Frequency [cycle/usecond]

## Analysis

recommendations and more



### High Memory Throughput

Memory is more heavily utilized than Compute: Look at the [Memory Workload Analysis](#) section to identify the DRAM bottleneck. Check memory replay (coalescing) metrics to make sure you're efficiently utilizing the bytes transferred. Also consider whether it is possible to do more work per memory access (kernel fusion) or whether there are values you can (re)compute.



### Roofline Analysis

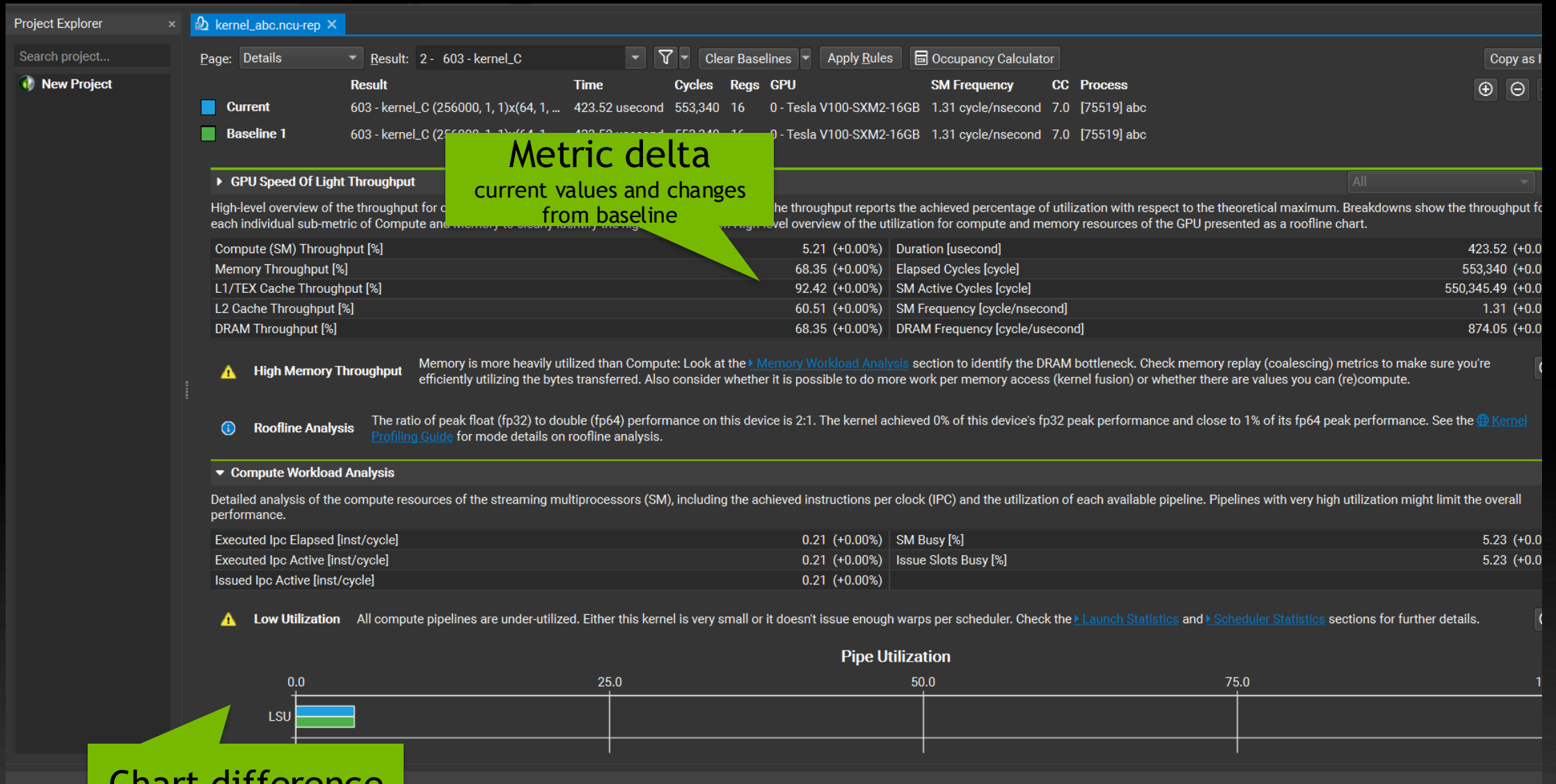
The ratio of peak float (fp32) to double (fp64) performance on this device is 2:1. The kernel achieved 0% of this device's fp32 peak performance and close to 1% of its fp64 peak performance. See the [Roofline Profiling Guide](#) for more details on roofline analysis.

### 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 Inst Elapsed Inst/cycle 0.21 | SM Busy [%]

# NSIGHT COMPUTE



Metric delta  
current values and changes  
from baseline

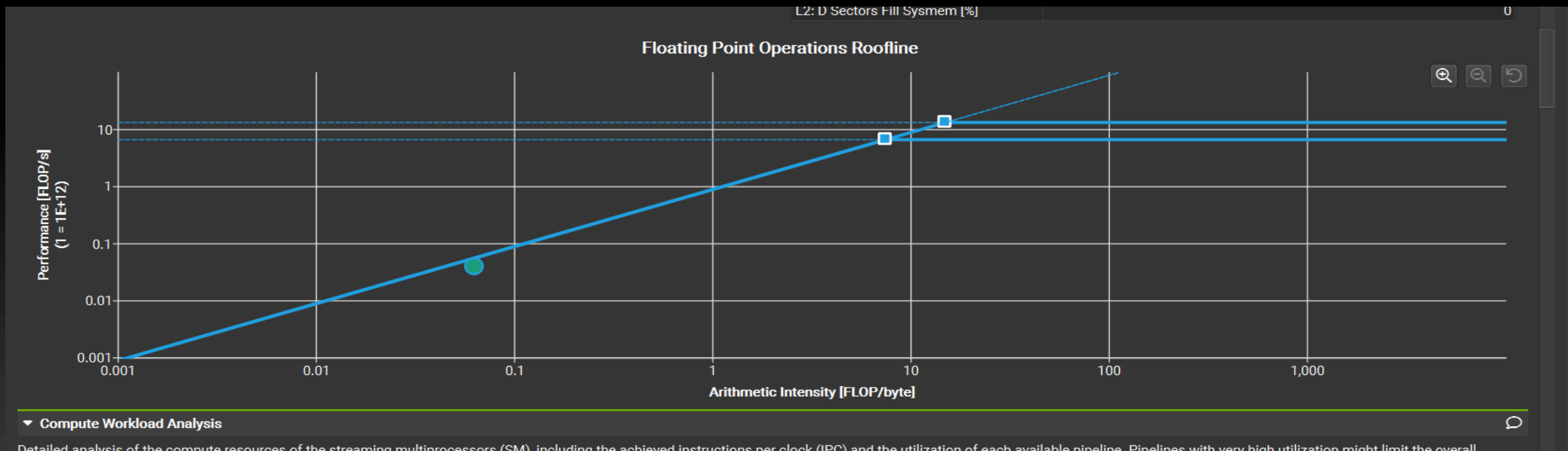
Chart difference  
current values and baseline  
values



# NSIGHT COMPUTE

## New Roofline analysis

- Efficient way to evaluate kernel characteristics, quickly understand potential directions for further improvements or existing limiters
- Inputs: Arithmetic Intensity (FLOPS/bytes)  
Performance (FLOPS/s)
- Ceilings: Peak Memory Bandwidth  
Peak FP32/FP64 Performance





**nVIDIA®**