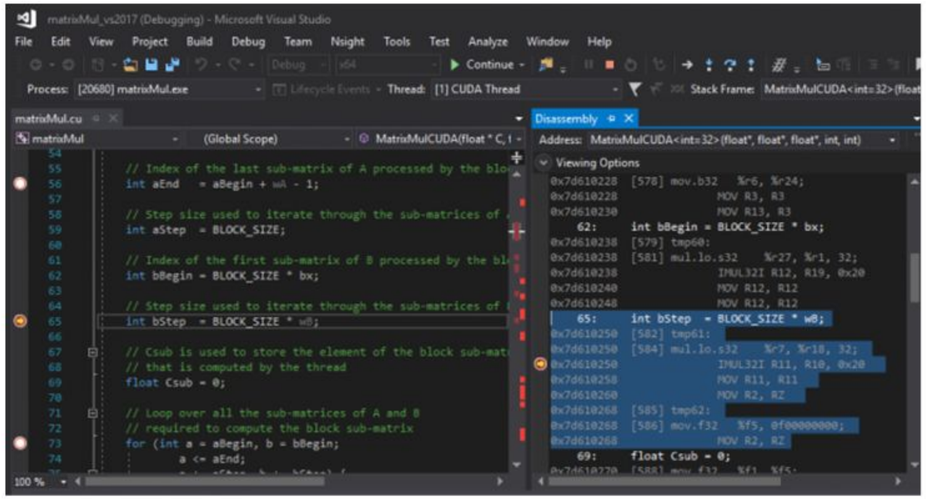
NVIDIA DevTools and HPC

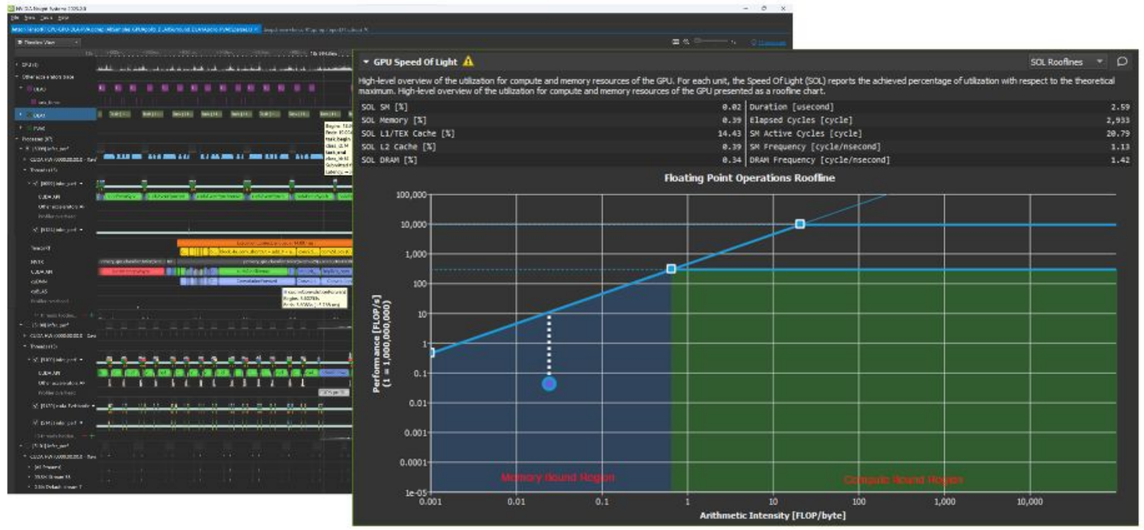
Holly Wilper, NVIDIA Developer Tools
Engineering Manager, Nsight Systems

Developer Tools Ecosystem

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



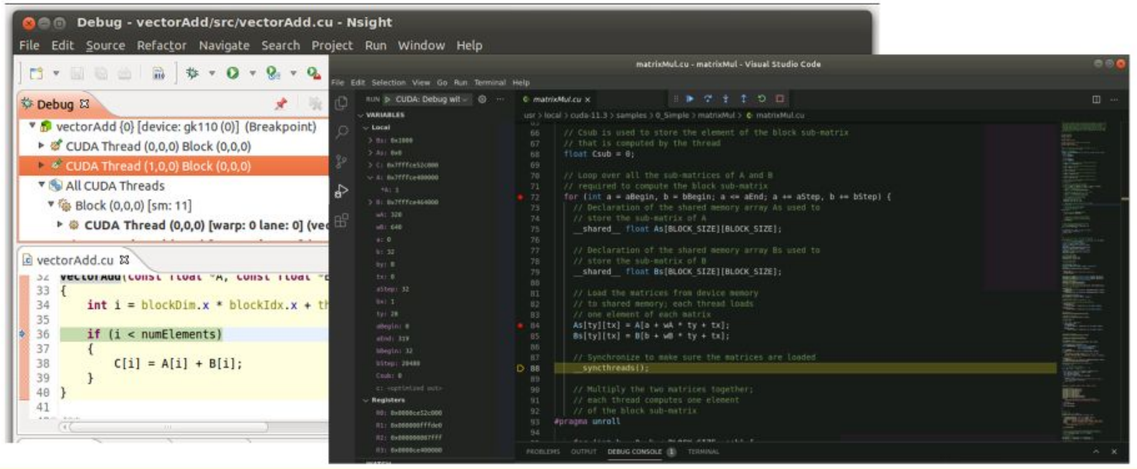
Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)



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



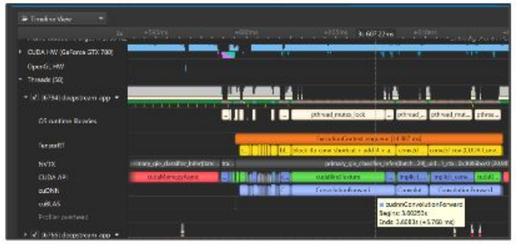
AI Integration: https://developer.nvidia.com/nsight-copilot

Nsight Profiler Product Family

Product roles

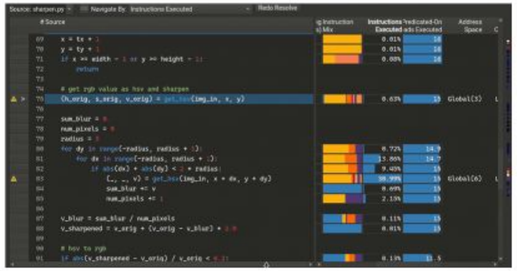
Nsight Systems

Analyze application algorithm system-wide



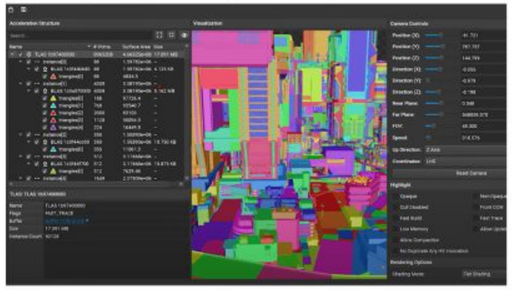
Nsight Compute

Debug/optimize CUDA kernels

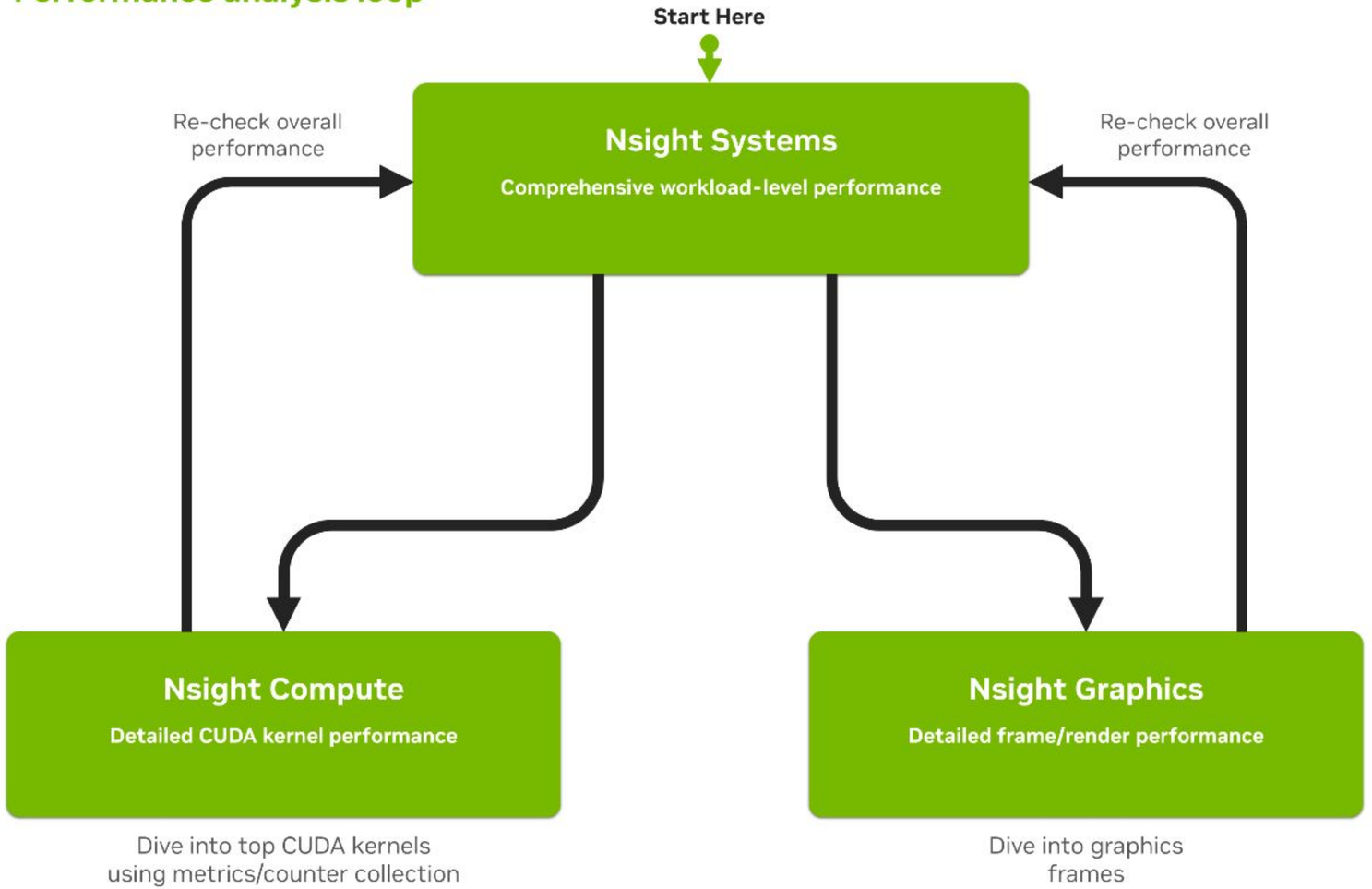


Nsight Graphics

Debug/optimize graphics workloads



Performance analysis loop

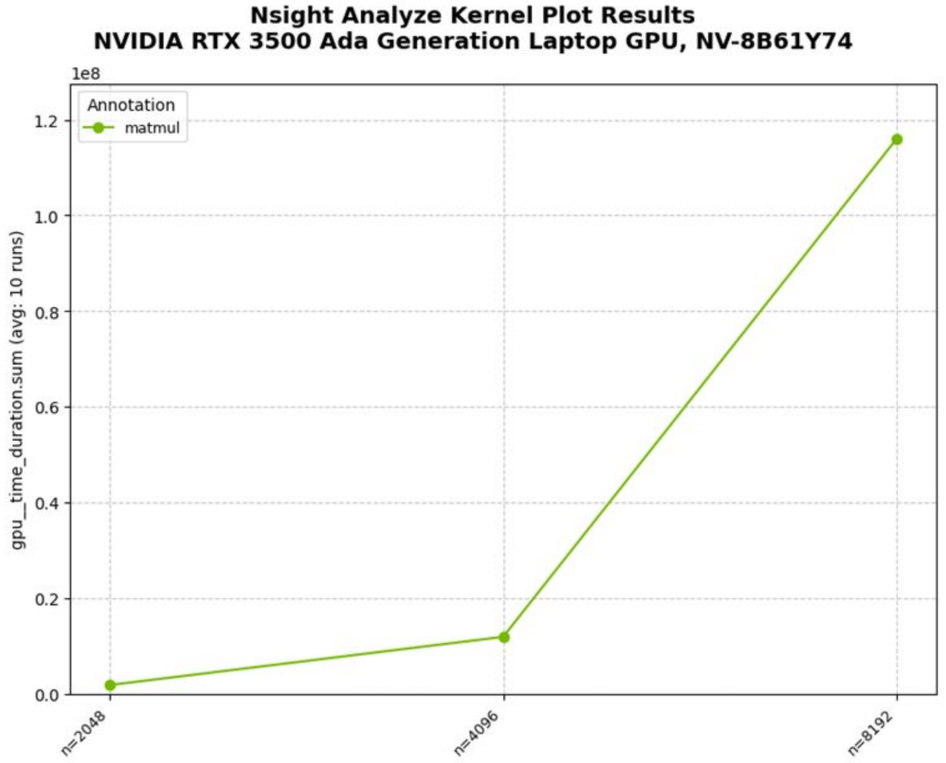


Introducing Nsight Python

Profiling directly from Python – no GUI required!

- Nsight Python now live!
- Python API for kernel profiling across multiple configurations
- Automatic plotting for visualization and comparisons
- Access profiling data directly from Python objects
- Available on PyPi and GitHub
  - https://pypi.org/project/nsight-python/
  - https://github.com/NVIDIA/nsight-python
- Will be extended to more tools use cases in the future (call for input!)

```python
@nsight.analyze.plot("02_parameter_sweep.png")
@nsight.analyze.kernel(configs=sizes, runs=10)
def benchmark_matmul_sizes(n: int) -> None:
    """
    Benchmark matrix multiplication across different sizes.
    The 'n' parameter comes from the configs list.
    """
    a = torch.randn(n, n, device="cuda")
    b = torch.randn(n, n, device="cuda")

    with nsight.annotate("matmul"):
        _ = a @ b
```



https://developer.nvidia.com/nsight-python

The Bugs Stop Here

Debug Numba CUDA for the First Time!

Numba CUDA provides a CUDA target for the Numba Python JIT Compiler

- GPU acceleration from Python

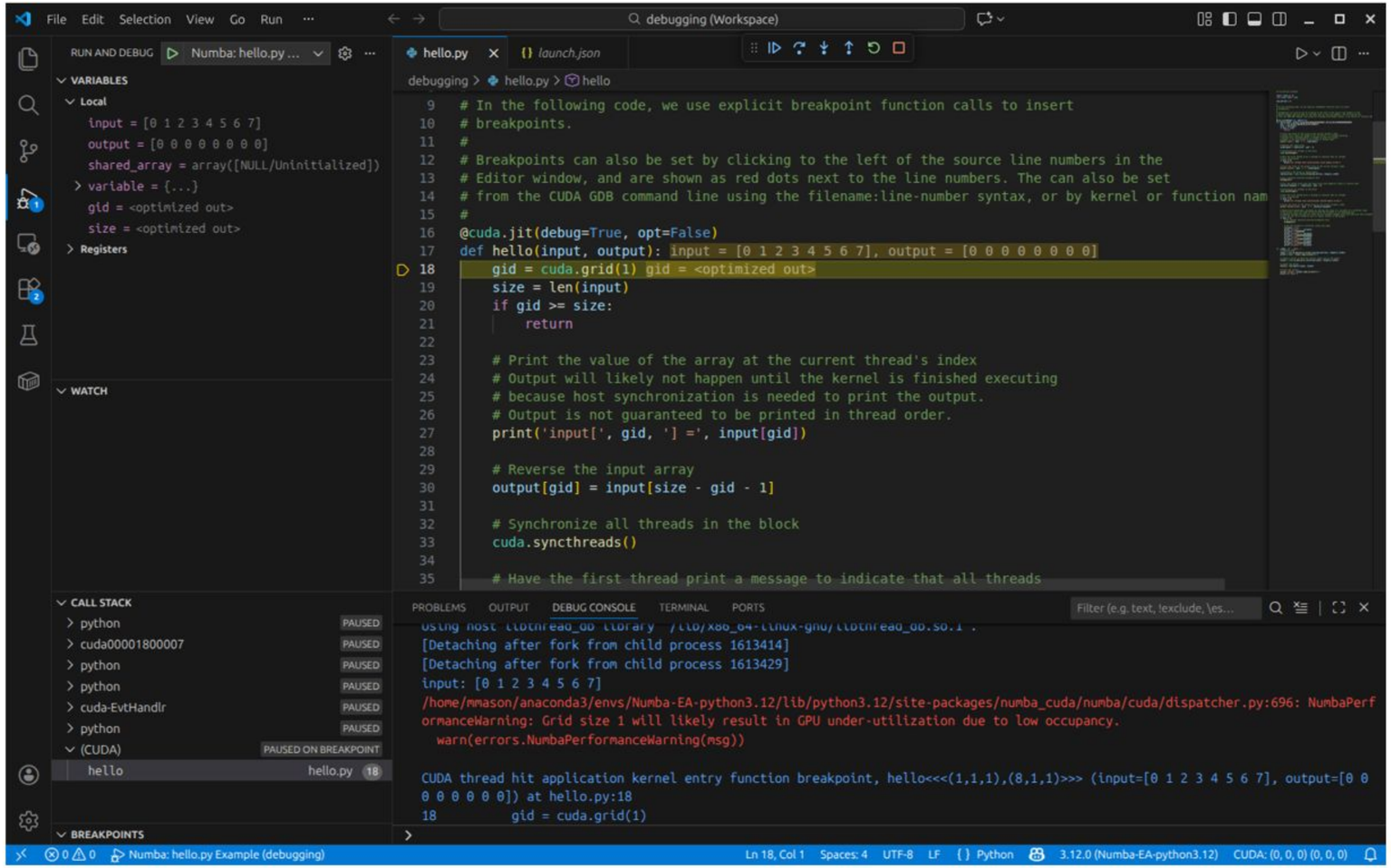
Enabled via **cuda-gdb** and **Nsight Visual Studio Code Edition**

Support for Python device code just like native debugging

Common Debug Tasks Supported

- Breakpoints
- Stepping
- Variable Inspection
- Etc.....

Numba CUDA users – we want to talk to you!



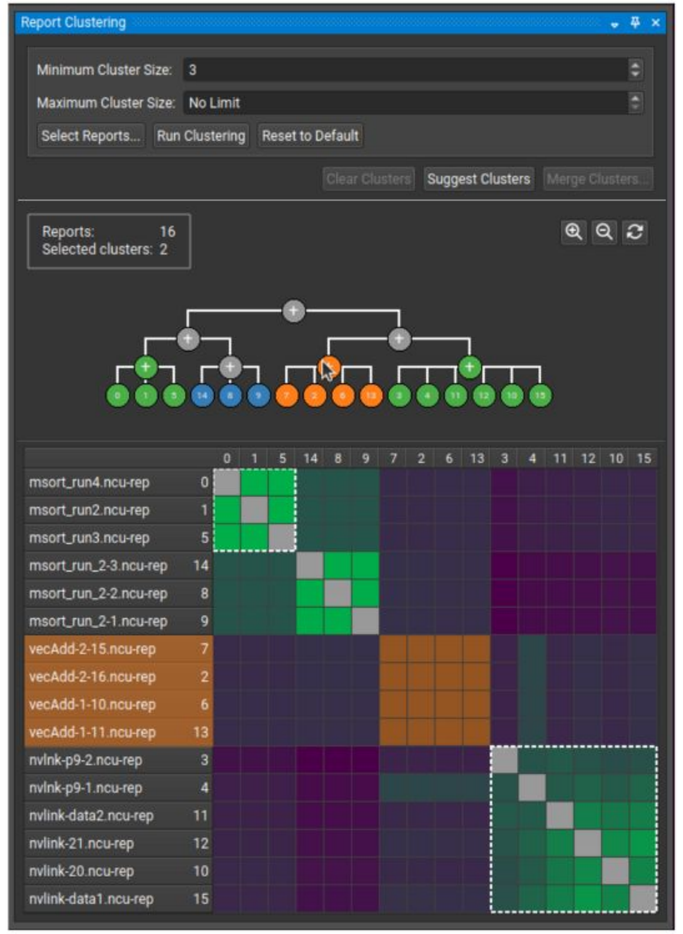
https://nvidia.github.io/numba-cuda/user/debugging.html

Making Your Kernels POP!

More details and automated anlysis with **Nsight Compute**

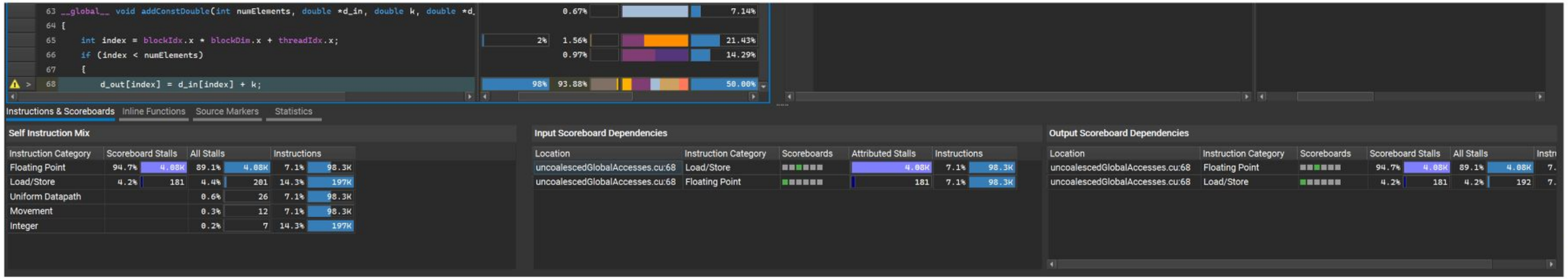
Multi-Report Analysis

- Automatic comparison of reports from:
  - Multiple processes, nodes, or configurations
- Intelligent report clustering and merging
  - Visually identify/cluster similar reports and outliers
  - Merge multiple reports (average, union, intersection)
- Save precious analysis time



Instruction Mix and Scoreboard Dependency Analysis

- Understand breakdown of source into instruction type (pipeline bottlenecks)
- Input and output dependencies causing stalls attributed to source
- “Who am I waiting on and who’s waiting for me?”



CUPTI Improvements

_v2 APIs - Beta: 13.2, Release: 13.3

- **User Defined Activity Records**
  - Less code maintenance (No bumping CUpti_ActivityKernel8 -> CUpti_ActivityKernel9 -> …)
  - Only select the fields you care about
    - Smaller activity records
    - Reduced backend overhead to collect unnecessary data
- **Multiple Subscriber Support**
  - Opportunistic multiplexing of CUPTI callback and activity APIs between multiple clients
  - Requires buy-in from first subscriber and all subscribers use _v2 APIs
  - Private record buffers per subscriber

CUPTI Python

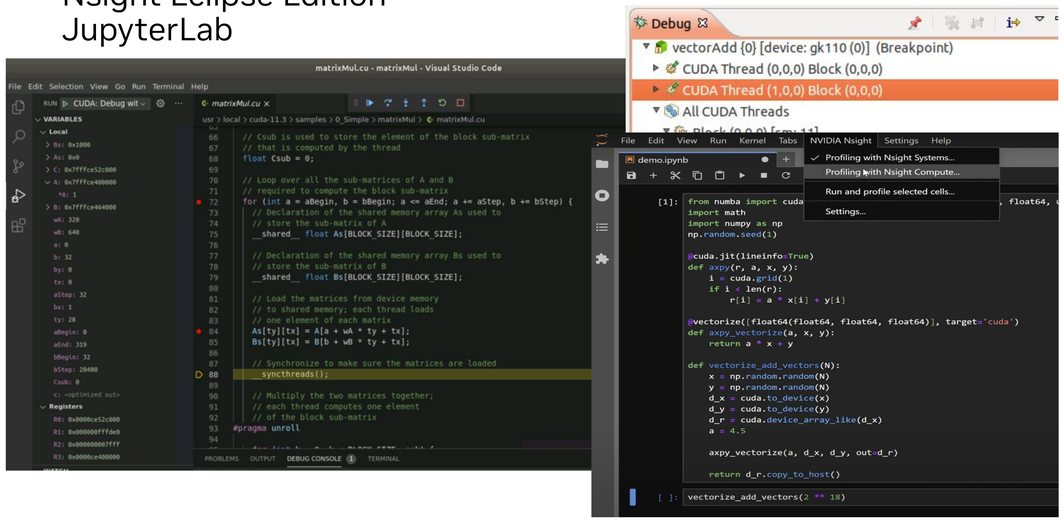
- Now supports PC Sampling and Range Profiling APIs

Hardware Event System

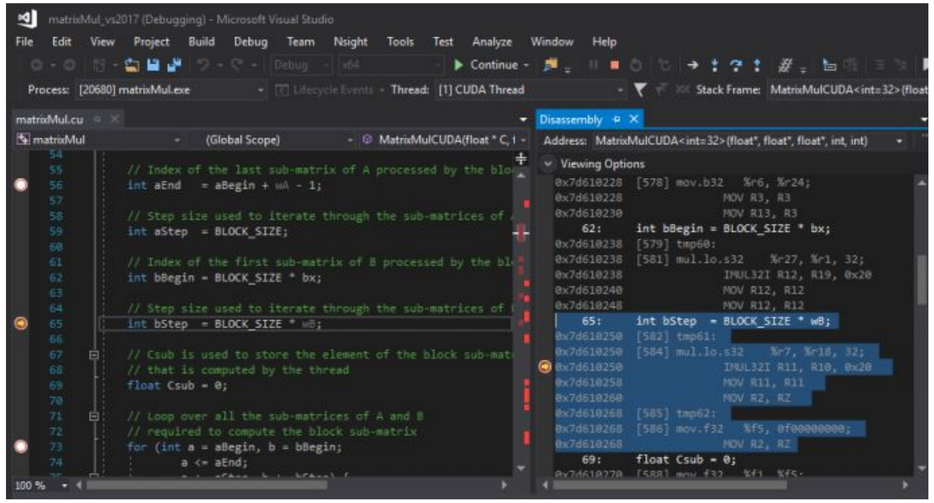
- cuptiActivityEnableHWTrace / CUPTI_ACTIVITY_ATTR_ENABLE_HES
- Blackwell+ hardware accelerated kernel start / stop timestamp collection

Developer Tools Ecosystem

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



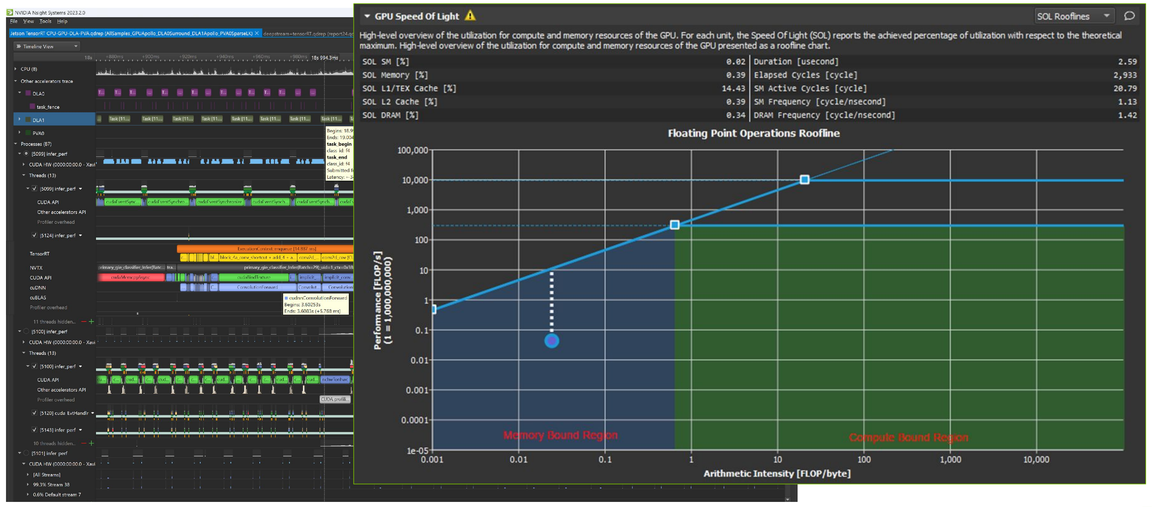
Debuggers: CUDA-GDB, Nsight Visual Studio Edition
Nsight Visual Studio **Code** Edition



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
```

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



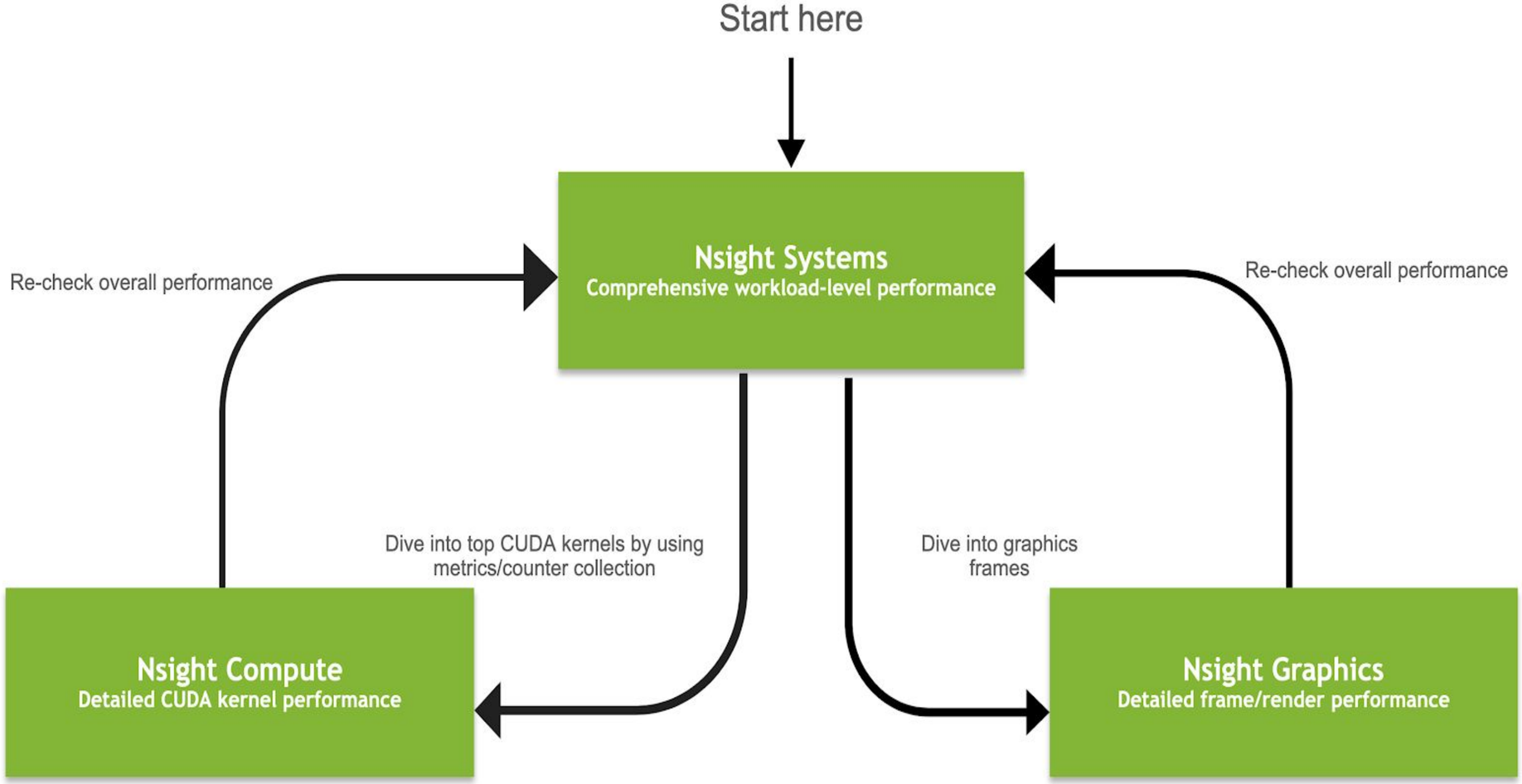
Nsight Product Family

Profiler Workflow

Nsight Systems -
Analyze application algorithm system-wide

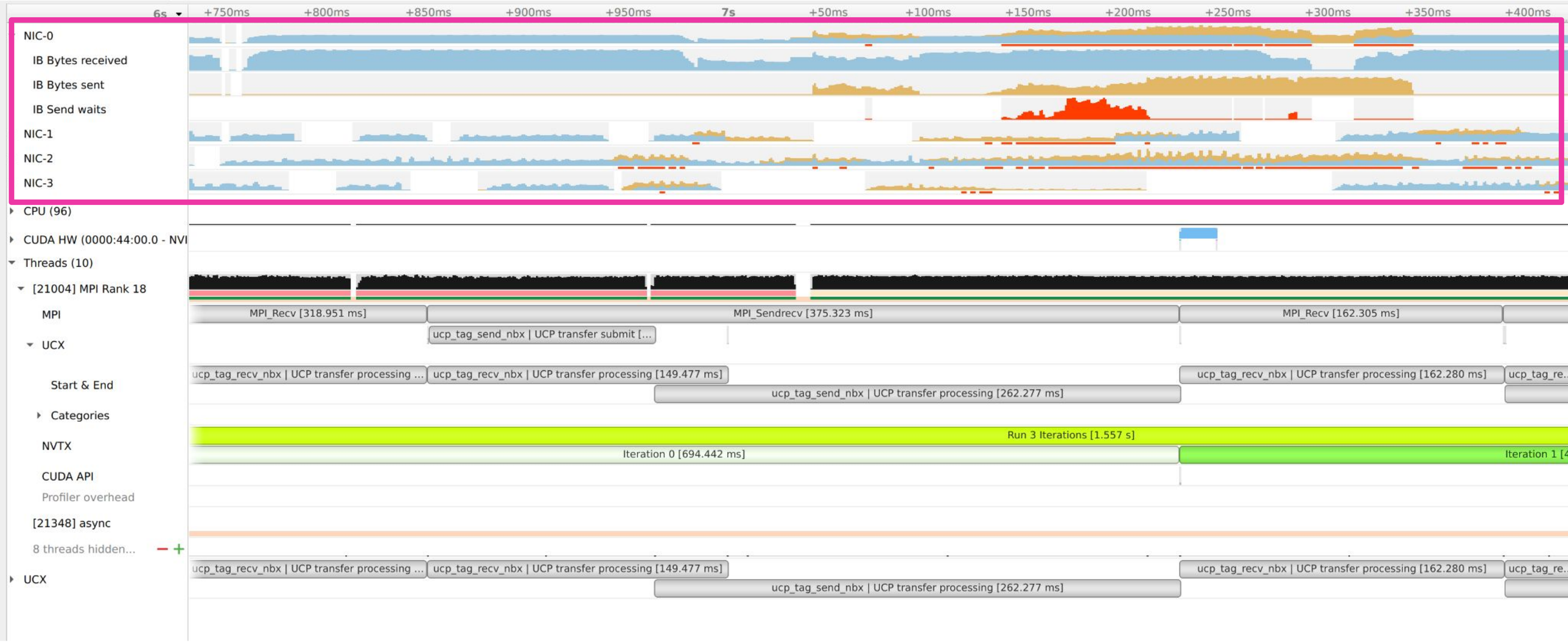
Nsight Compute -
Debug/optimize CUDA kernel

Nsight Graphics -
Debug/optimize graphics workloads



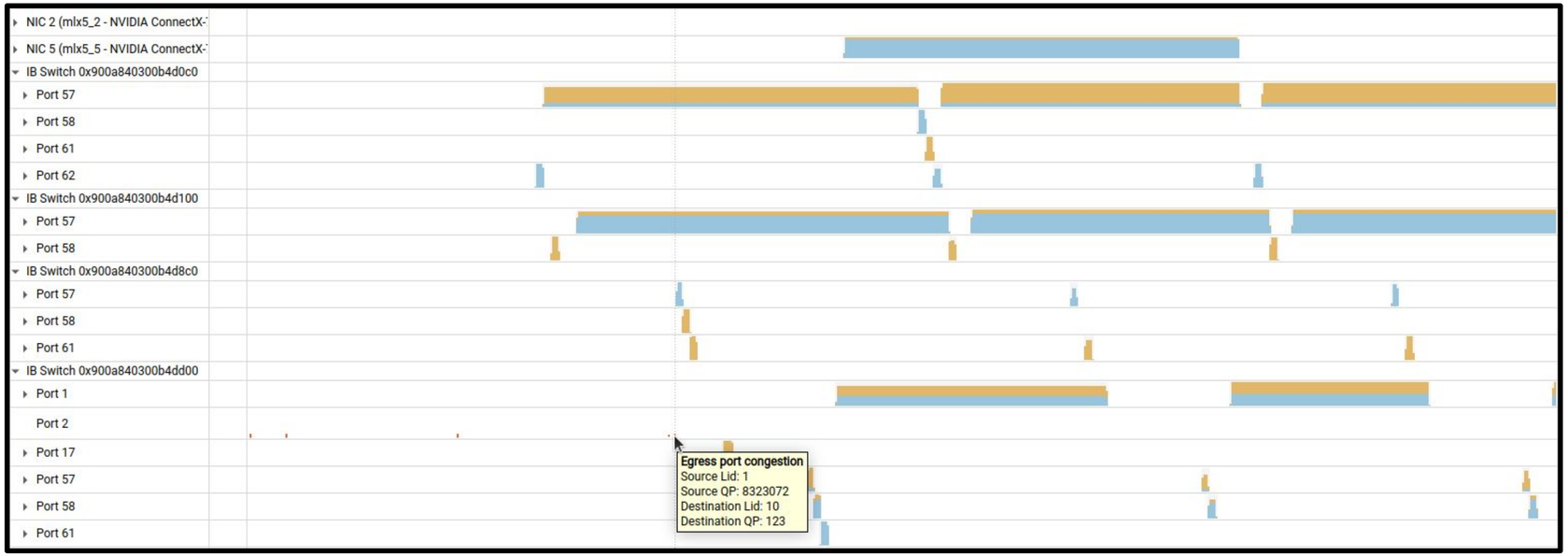
Nsight Systems Network Analysis

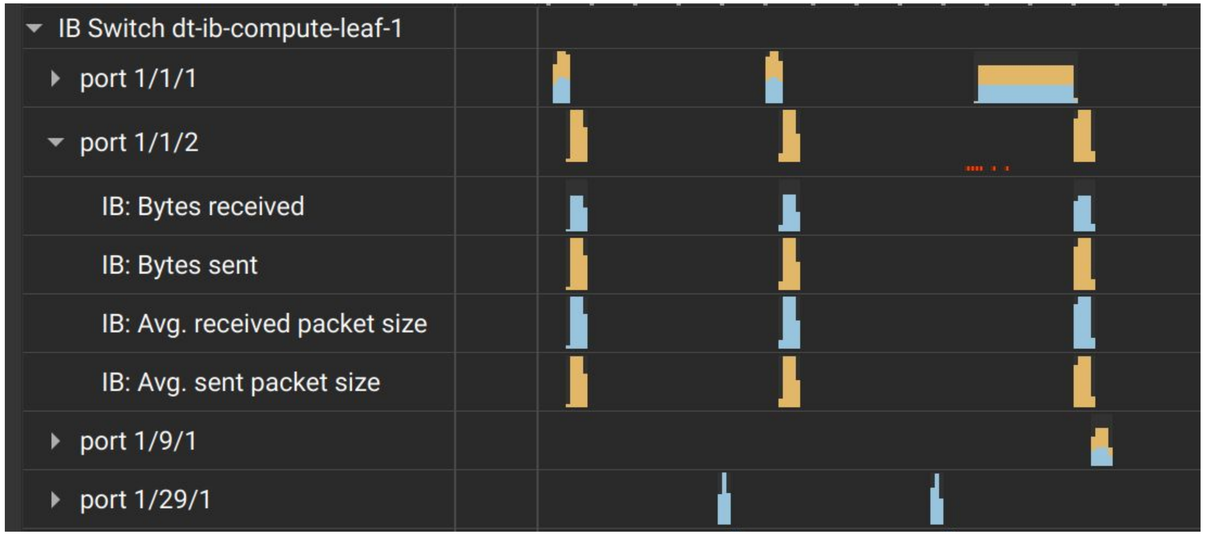
NVIDIA NIC/HCA Metrics Sampling (InfiniBand & RoCE)

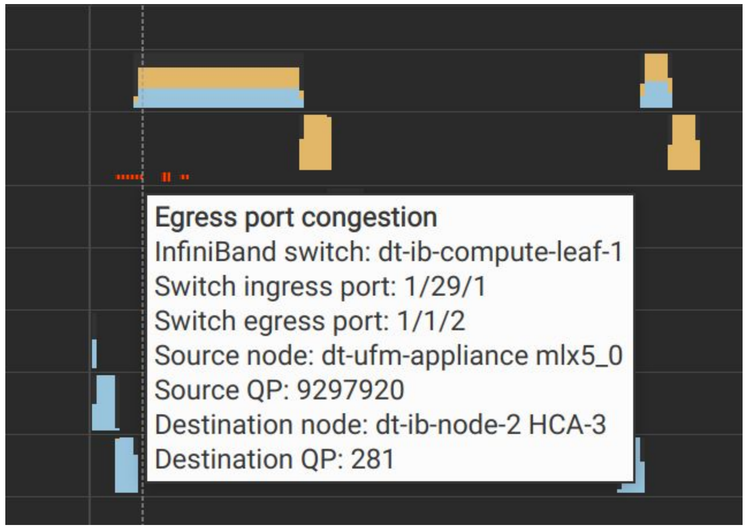


NVIDIA Quantum-2 Switch Sampling

Send, Receive, Congestion, & Now Per-Port Sampling

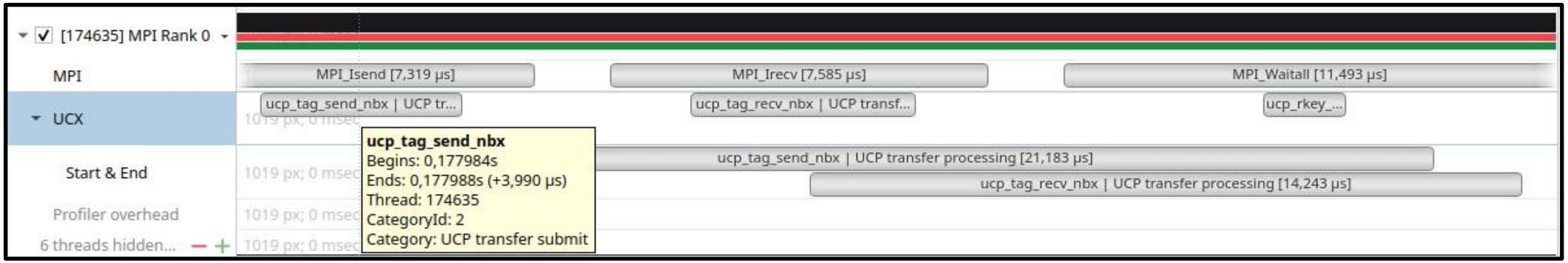


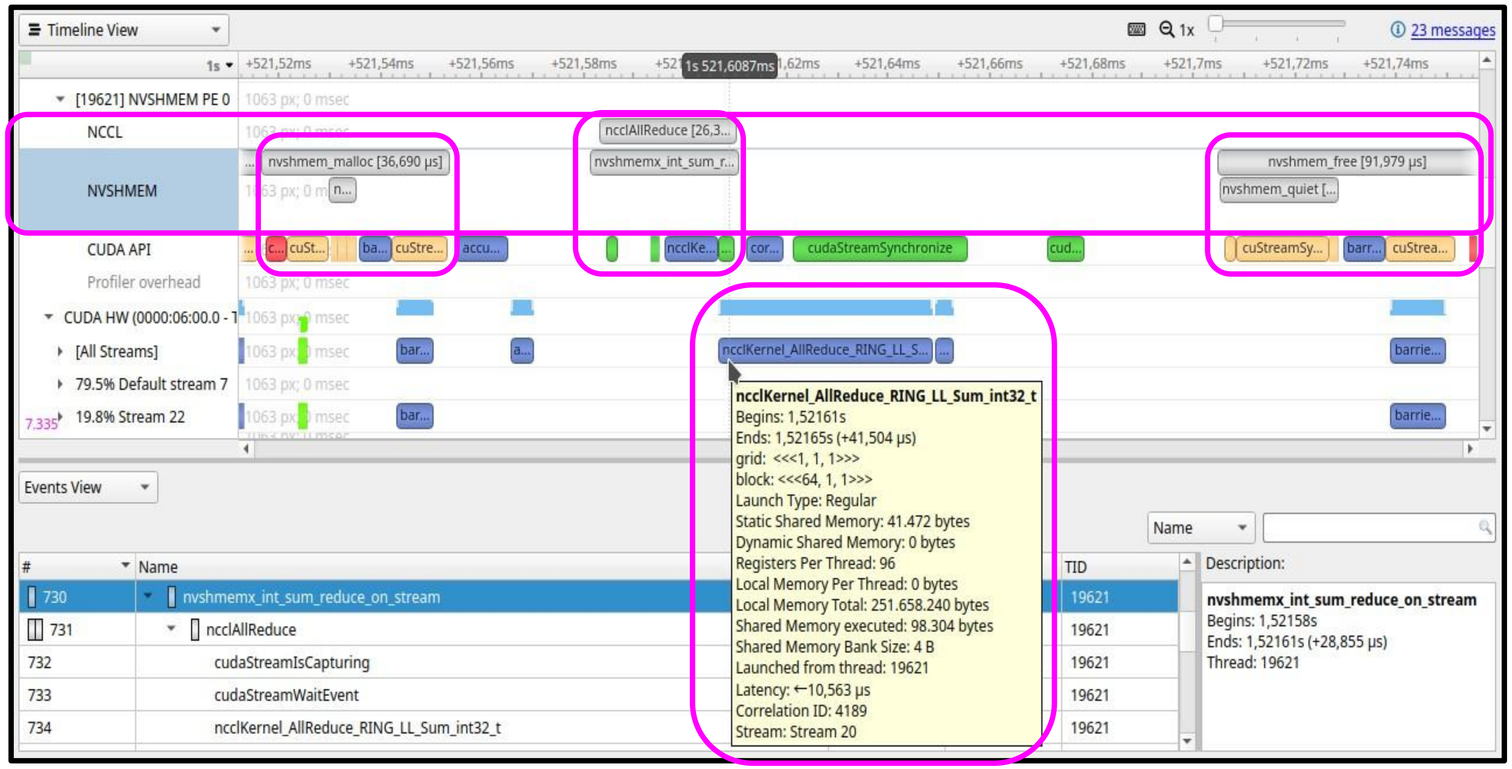




Communications Trace - LIBC, UCX, MPI, NCCL, & NVSHMEM

APIs, Destinations, Sizes, Async Completion, & Related GPU Events



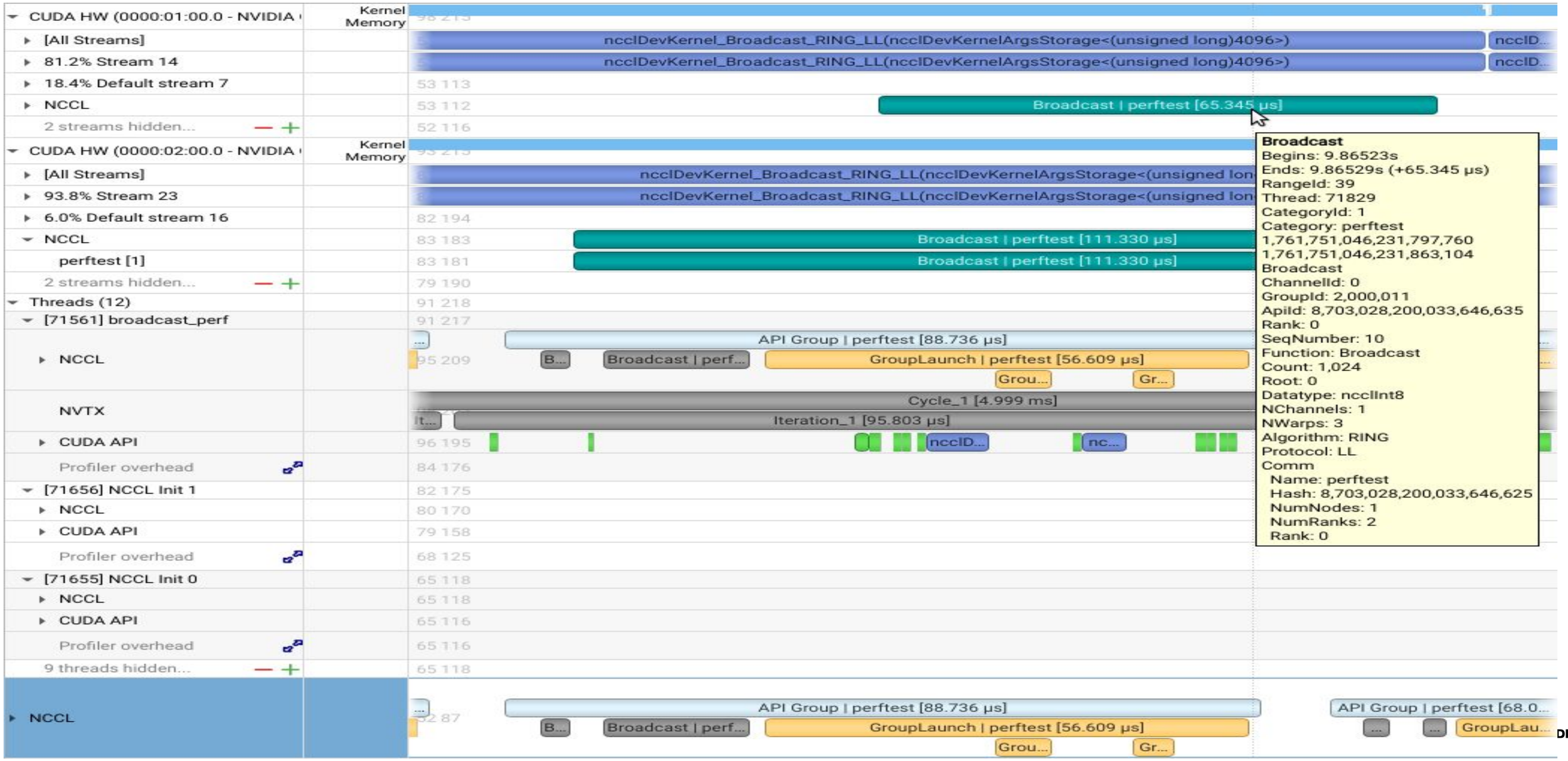


Description:

MPI_Bcast
Begins: 0,164935s
Ends: 0,165045s (+109,210 µs)
Thread: 1342434
Bytes sent: 0
Bytes received: 4
Root: 0
MPI_COMM_WORLD

Advanced NCCL Tracing

Use -t nccl option

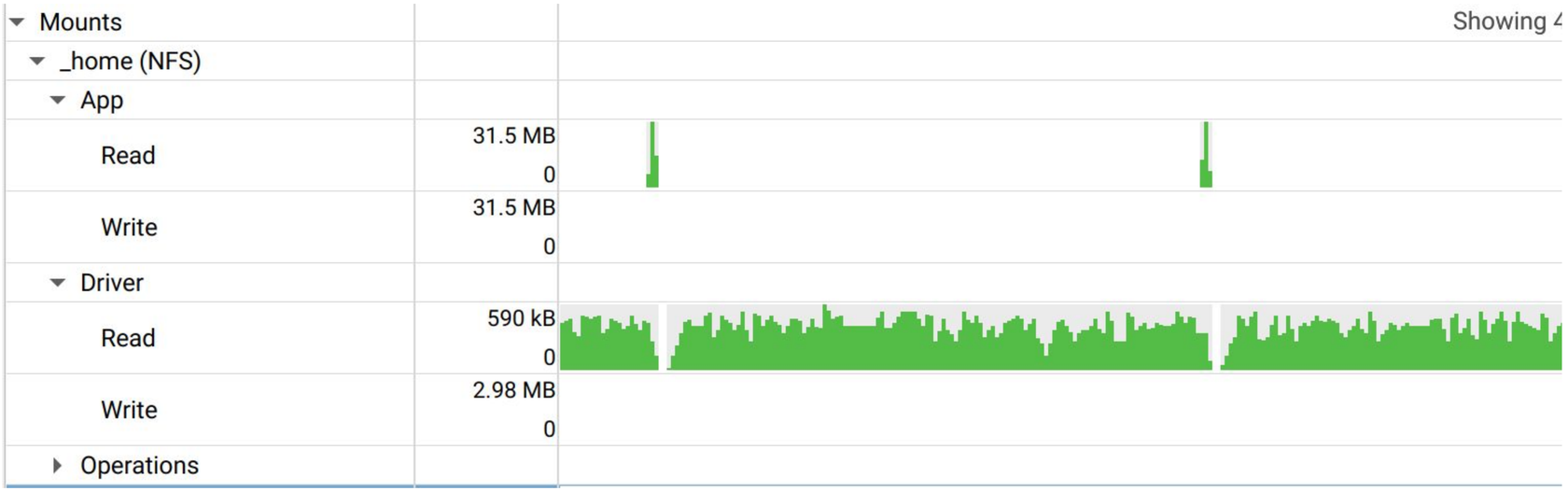


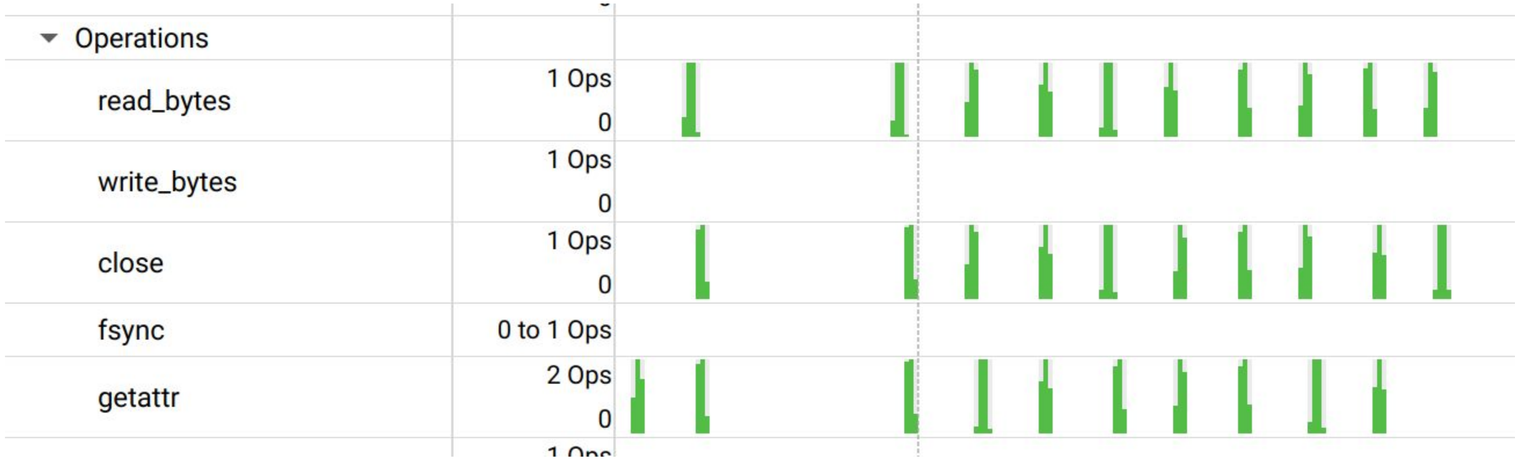
Lustre performance counters

Display Lustre performance counters & operations count

NFS performance counters

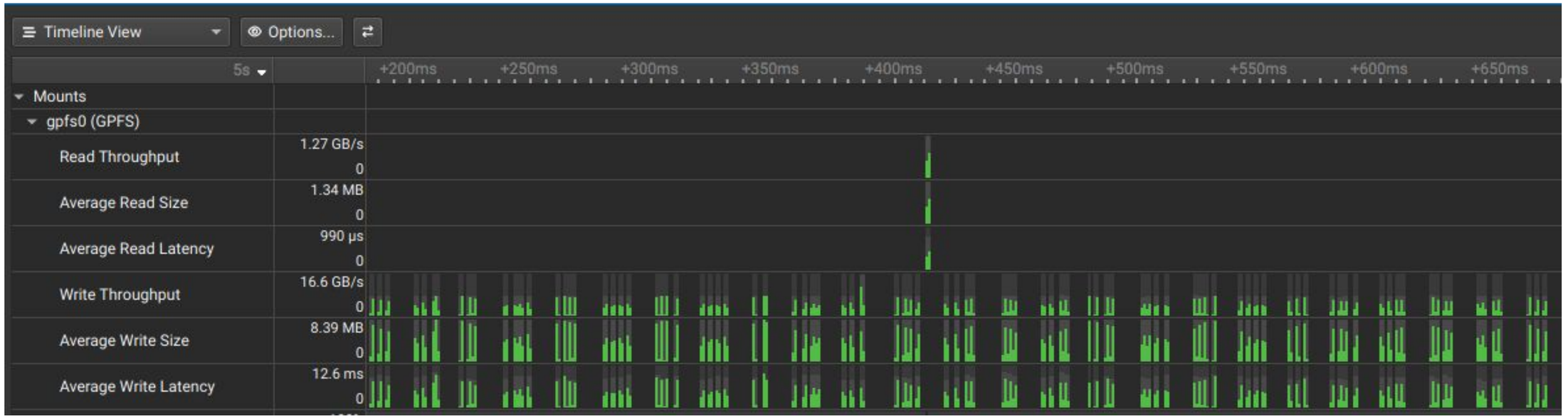
Display NFS performance counters & operations count





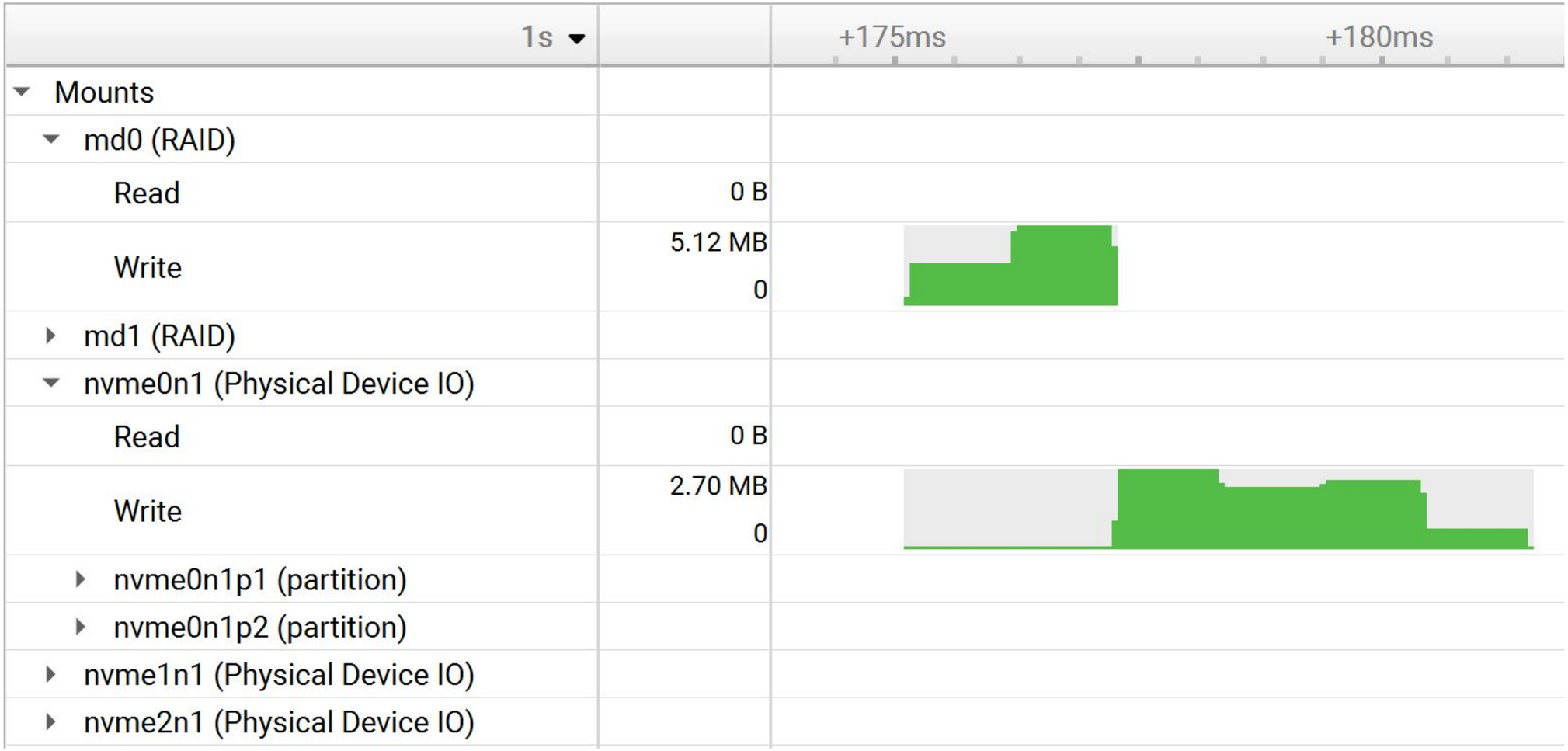
GPFS performance counters

Display GPFS client volumes performance counters



Local disks & NVMe-oF

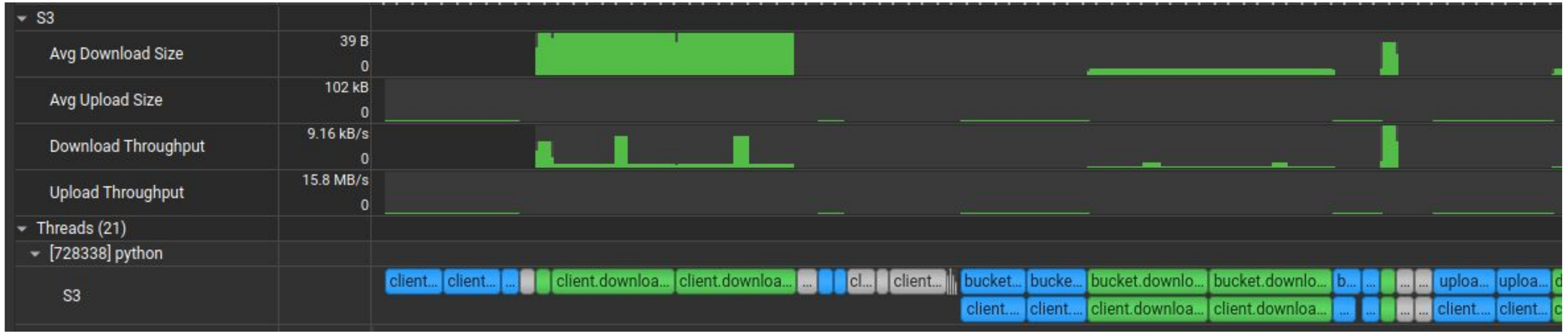
Display local disks and NVMe-oF performance counters



S3 Trace + Counters

Trace common S3 communication libraries

Display S3 throughput counters



NVTX and the Plugin System

NVIDIA Tools Extensions (NVTX)

C, C++, Python, & Rust

Decorate application source code with annotations (markers, ranges, nested ranges, …)
Help visualize execution with debugging, tracing and profiling tools

Header-only library: https://github.com/NVIDIA/NVTX/tree/release-v3/

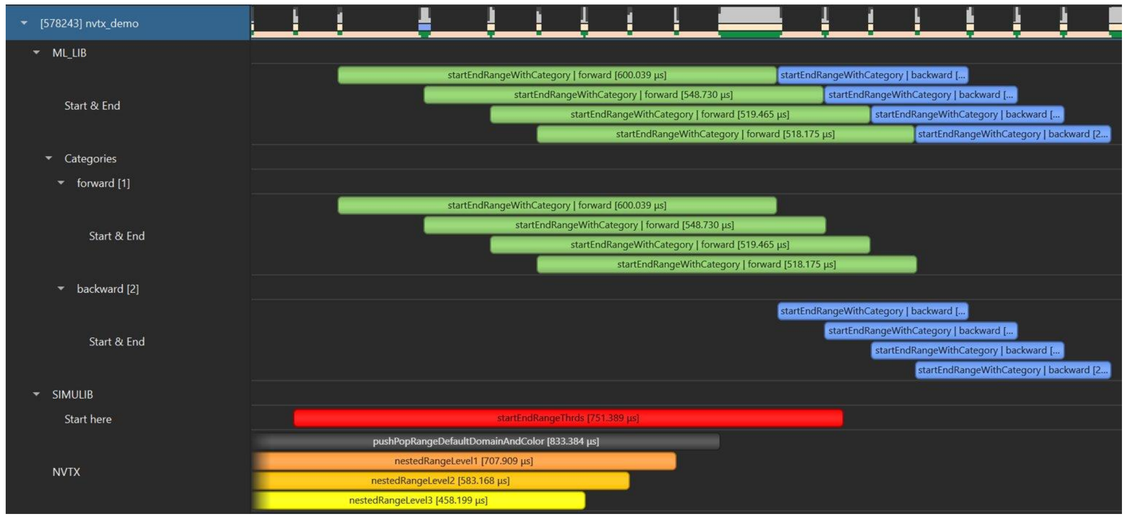
```
#include <nvtx3/nvToolsExt.h>
```

Marker:

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

Push-Pop range:

```
nvtxRangePush("This is a push/pop range");
// Do something interesting in the range
nvtxRangePop();
```



Start-End range:

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

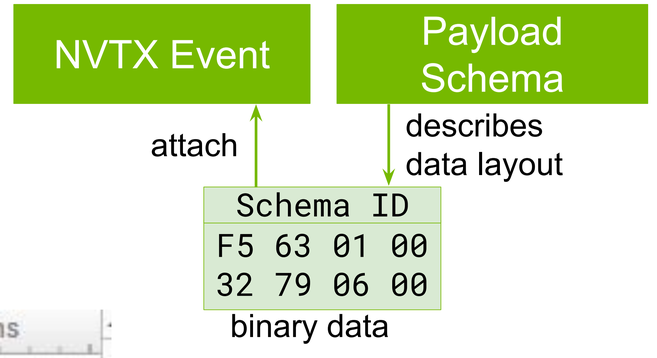
Advanced concepts like: colors, domain, category, registered strings, …

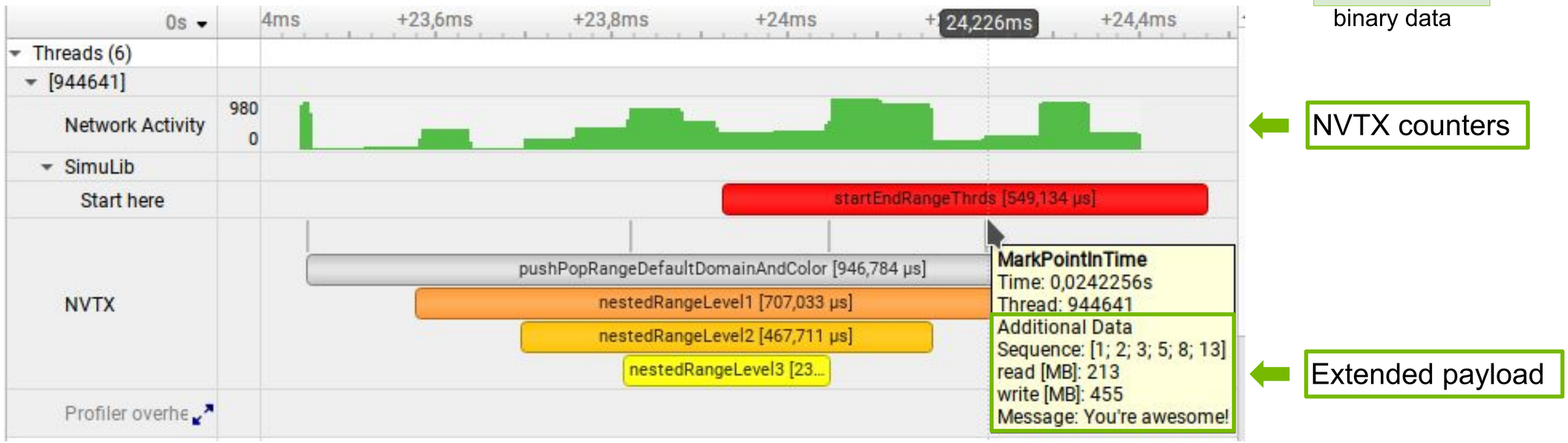
API references: https://nvidia.github.io/NVTX/doxygen/index.html and https://nvidia.github.io/NVTX/doxygen-cpp/index.html

NVTX Extended Payloads & NVTX Counters

Increase the flexibility of NVTX annotations

- Pass arbitrary data to NVTX events and allow users to define the layout of this data without additional data conversion
- NVTX markers, push/pop ranges and start/end ranges with extended payload
- NVTX counters
  - immediate and deferred counter samples
  - Data layout of counter groups via extended payload schemas
  - User-defined scopes





NVTX counters

Extended payload

NVTX Writer API

- Allows third-party app to use an NVTX-like API to export event data as Nsight Systems report file
- Supports extended payloads, so app-specific C-style struct data can be included in events as binary blobs

```
typedef struct
{
  uint64_t time_start;
  uint64_t time_end;
  uint32_t task_id;
  uint32_t color;
  uint8_t  priority;
} event_data;
```

```
nvtxPayloadSchemaEntry_t schema[] =
{
  {TIMESTAMP | RANGE_BEGIN, TYPE_UINT64,     "Time Start"},
  {TIMESTAMP | RANGE_END,   TYPE_UINT64,     "Time End"},
  {0,                       TYPE_UINT32,     "Task ID"},
  {0,                       TYPE_NVTX_COLOR, "Color"},
  {0,                       TYPE_UINT8,      "Priority"}
};

nvtxPayloadSchemaAttr_t schema_attr =
  {..., "Event Data", ..., schema, ...};
```

1) Static data layout description* can be put into a header.

```
uint64_t schemaId = nvtxwSchemaRegister(stream, schema_attr);
```

2) Register the data layout (once).

```
event_data event = {100000000, 150000000, 43322, 0xFFFF0000, 9};
nvtxPayloadData_t payload = {schemaId, &event, sizeof(event)};
nvtxwEventWrite(stream, &payload, 1);
```

3) Write payload to output stream

* macro names have been shortened to fit on the slide.

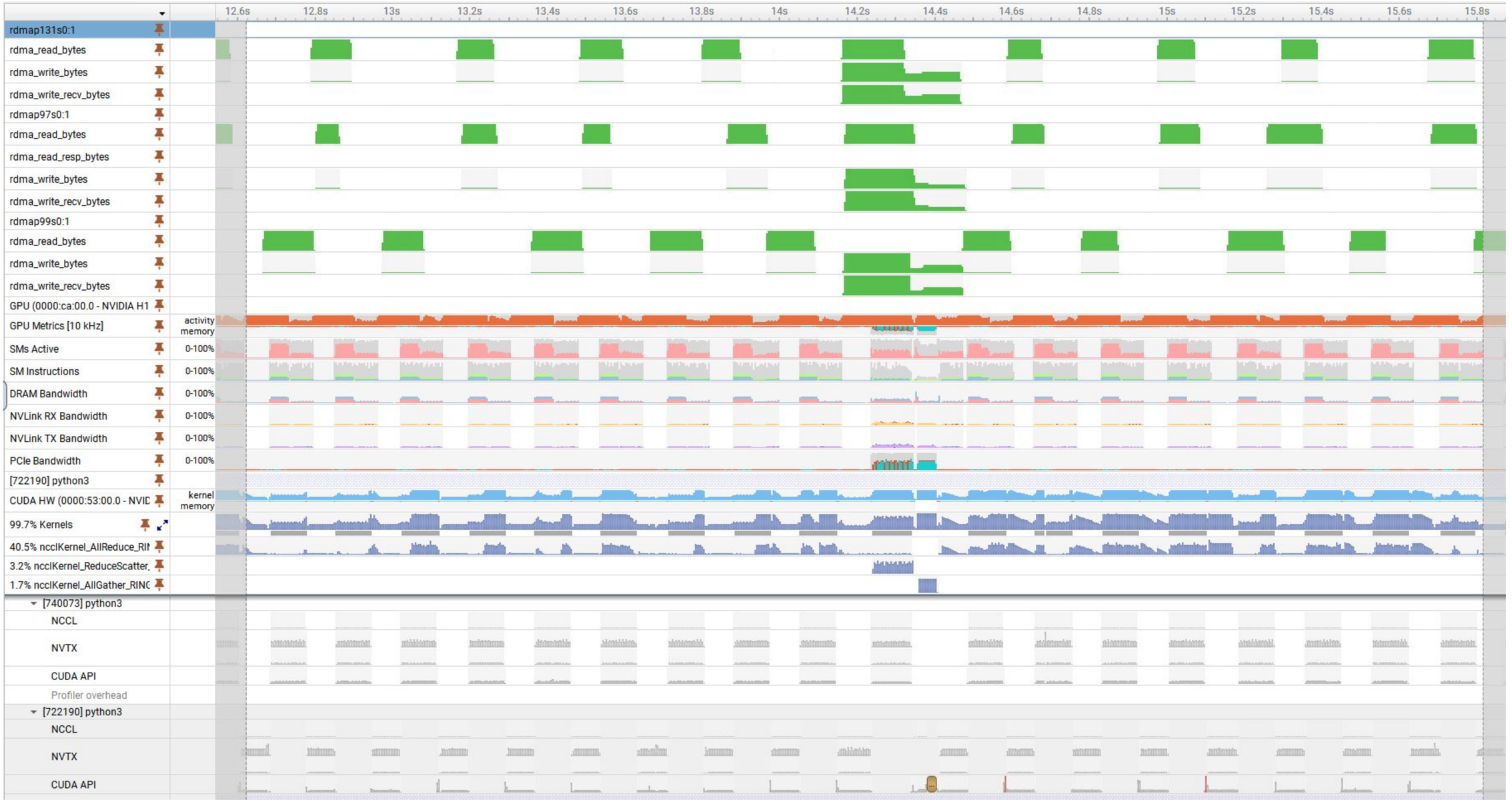
Nsight Systems Plugins

```
atrachenko-dem1 /opt/nvidia/nsight-systems-cli/2024.3.1/target-linux-x64/plugins/efa_metrics %
atrachenko-dem1 /opt/nvidia/nsight-systems-cli/2024.3.1/target-linux-x64/plugins/efa_metrics % ll
total 2468
-rwxrwxr-x 1 root root 2520864 Apr 17 07:31 nic_sampler
-rw-rw-r-- 1 root root     959 Apr 17 07:31 nsys-plugin.yaml
atrachenko-dem1 /opt/nvidia/nsight-systems-cli/2024.3.1/target-linux-x64/plugins/efa_metrics % cat nsys-plugin.yaml
PluginName: efa_metrics
ExecutablePath: nic_sampler
Description: Collects AWS EFA Infiniband and Ethernet metrics
ExtendedDescription: |
  Usage and available arguments:
    -efa                Sample all AWS EFAs (default true)
    -efa-non-rdma       Sample Infiniband non-RDMA counters
    -efa-work-requests  Sample Infiniband WorkRequest counters
    -errors             Sample error counters
    -eth <string>       Comma separate list of net adapter names seen in /sys/class/net/
    -freq <int>         Target sample frequency in hertz (default 10)
    -mode <string>      Pick marks or counters (default "marks")
    -packets            Sample packet counters
    -print              Print samples to StdOut, not just submit NVTX
    -struct             Sample as structs rather than individual fields

  If -eth=all the tool will attempt to filter only physical adapters.
  -mode=counters is required to activate NVTX Counters (to produce graphs).

atrachenko-dem1 /opt/nvidia/nsight-systems-cli/2024.3.1/target-linux-x64/plugins/efa_metrics % nsys profile --enable efa_metrics,-mode,counters,-eth,enp0s31f6 sleep 10
```

- Take your process to collect data and add NVTX to provide data to Nsight Systems
- Add that process into the Nsight Systems plugin directory
- Write a plugin manifest file
- Now you can call your collection process directly from the Nsight Systems CLI
  - --enable name,flag1,arg2=val1,arg3=val2,...,valN
  - Flags and arguments are passed to your process when started

https://nvidia.github.io/NsightSystemsPlugins/

Use Case: Amazon AWS Elastic Fabric Adapter(EFA) Sampling



Nsight Systems AI Preview

What Is the Nsight Systems AI Skill?

A capability pack for AI-assisted performance analysis

- Included with the **Nsight Systems installation**
- **Agent-agnostic**
- Connects **AI agents** to Nsight Systems knowledge, tools, and workflows
- Uses a central **SKILL.md** file to guide the agent

NOTE: This is a preview feature. Future release will extend these capabilities.

What Can the Nsight Systems AI Skill Do?

Support the workflow from data collection to root-cause analysis

01
Run a trace on a target application.

02
Inspect report data and extract supporting evidence.

03
Discover the supported Nsight Systems recipes and run the appropriate recipe for the task.

04
Launch specialized workflows to identify stutter frames and their likely causes.

Takeaway: The skill helps the agent select and run the appropriate capability or recipe for the current task.

The Nsight Systems Skill saves time and tokens

The skill is architected with Progressive Discoverability

- Starts with the central **SKILL.md** file.
- Gradually discovers the method relevant to the requested task.
- Avoids loading the entire documentation set at once, preventing context bloat.
- Uses fewer context tokens, reduces inference cost, and minimizes execution latency.

Key idea: The agent discovers capabilities step by step and loads only the information required for the task.

THANK YOU!

Download https://developer.nvidia.com/nsight-systems

NOTE: Website versions newer than CUDA Toolkit

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

Email nsight-systems@nvidia.com

Documentation https://docs.nvidia.com/nsight-systems

- **Self-paced course:** Optimizing CUDA Machine Learning Codes With Nsight Profiling Tools
- **Blog:** Optimizing CUDA Memory Transfers with NVIDIA Nsight Systems

Nsight Systems Overview

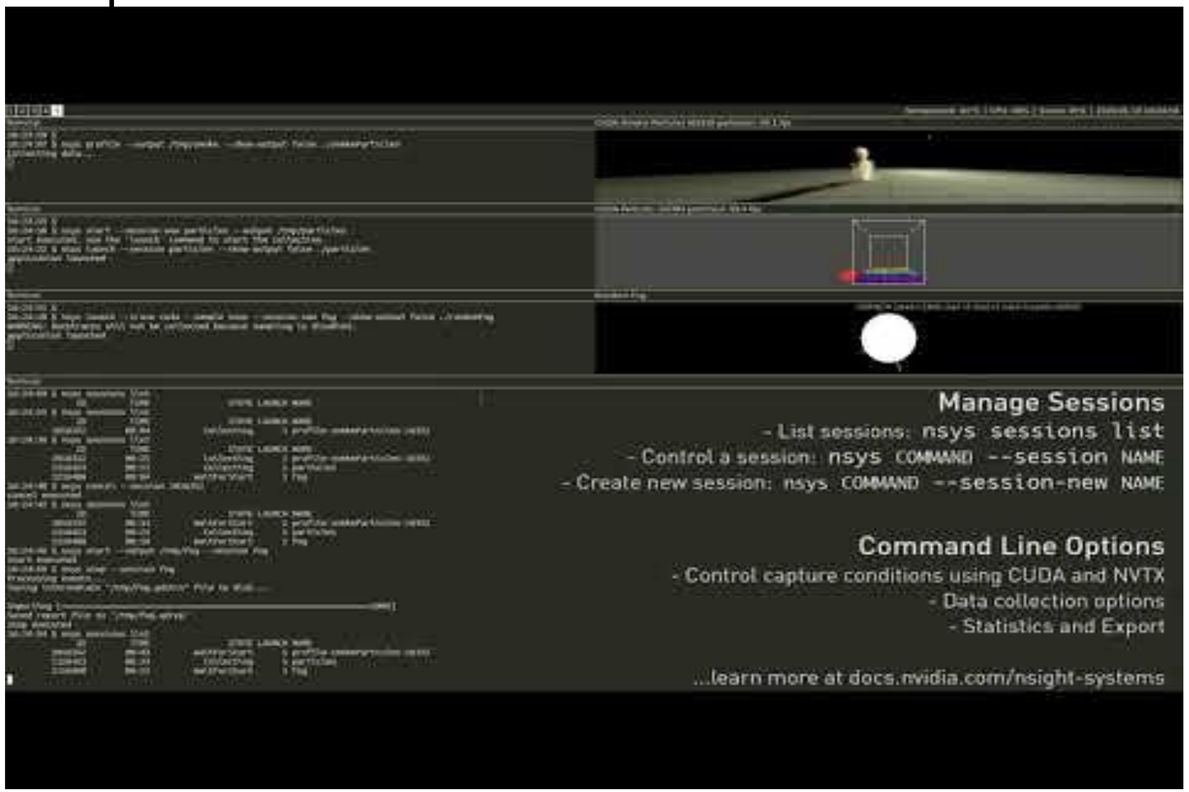
- System-wide application algorithm tuning
- Locate optimization opportunities
    - Visualize millions of events on a very fast GUI timeline
    - See gaps of unused CPU and GPU time
- Balance your workload across multiple CPUs and GPUs
    - CPU algorithms, utilization, and thread state
    - GPU streams, kernels, memory transfers, etc
    - NICs and Switch counters and congestion control
- Multi-platform: Linux & Windows, x86-64, ARM server
    - Mac (host-only)
    - Tegra - Linux & QNX
- Advanced statistical analysis system

Timeline Features

- API Trace
  - CUDA 10+ API & GPU workload ranges & mem transfers with correlation
  - Libraries: cuBLAS, cuDNN, cuDF, TensorRT
  - OpenACC, DirectML
  - Direct3D12, DXR, Direct3D11, WDDM, Vulkan, OpenGL
  - Communication APIs (MPI, etc)
- OS
  - Thread state and CPU utilization
  - OS runtime trace
  - ftrace or ETW (page faults, signal, interrupts, ...)
- User Annotations APIs
  - NVTX, PIX, Vulkan debug markers, KHR_debug

Other Key Features

- Thread call-stack periodic sampling
  - Backtraces via hardware LBRs or frame pointers
  - Hot functions
- Command Line Interface (CLI)
  - No host PC required to record
  - Works in containers & VMs
  - Scriptable / interactive mode
  - Multiple sessions
  - Multiple reports per launch
- Analysis
  - Recipes, statistics, expert systems
  - Data access: sqlite, arrow, parquet, json, …

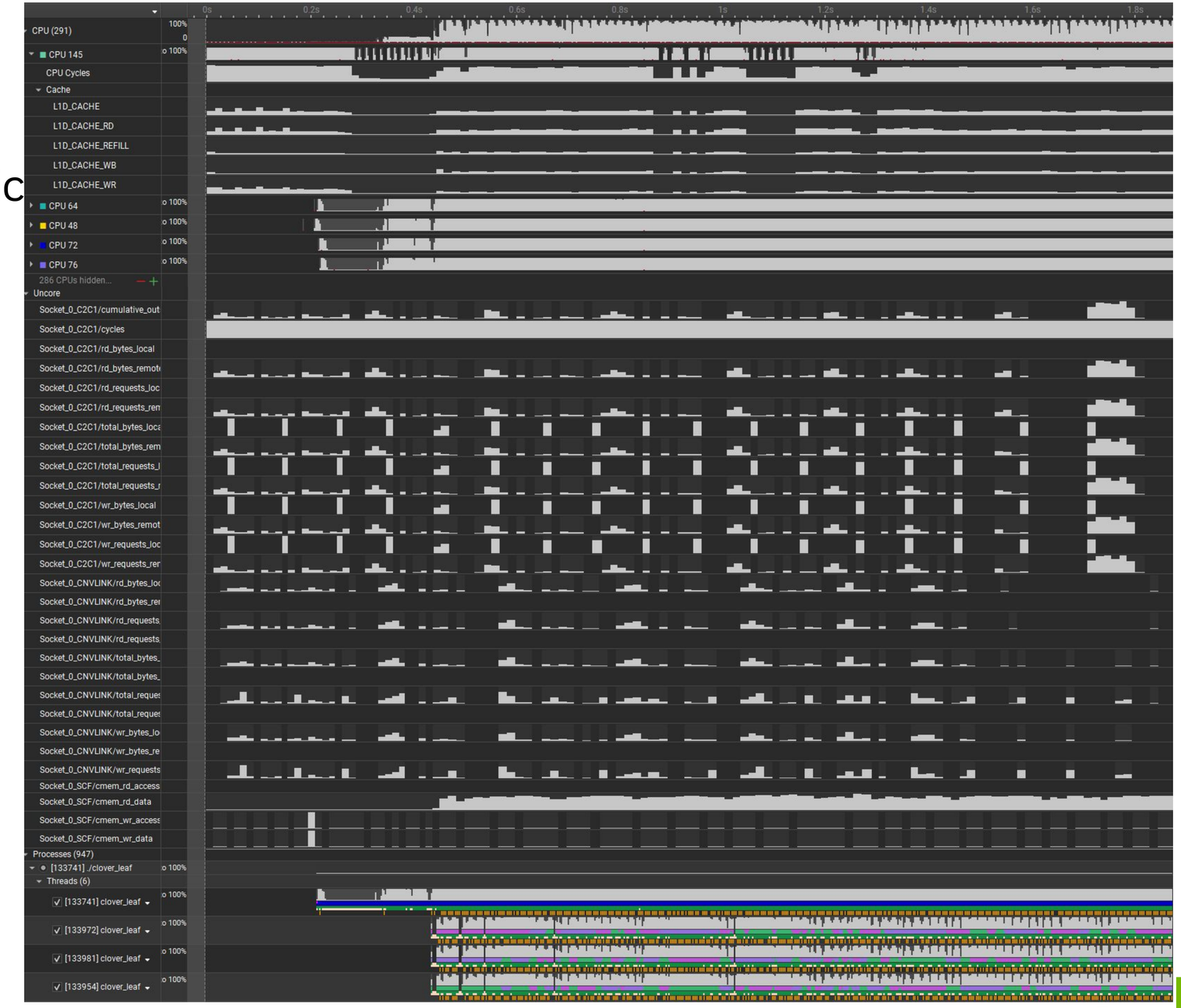


CPU Performance Counters Sampling

Sampling Grace counters & metric

See data from

- Cores
- Uncore
- Coherency
- Caches
- Buses
  - Memory
  - NVLink-C2C
- …



GPU Metrics Sampling (Compute & Graphics)

Ex: Multi-Node Deep Learning Training

Multi-Node / Multi-Report Analysis

Scale analysis to thousands of reports
  On your workstation or across a cluster or cloud

Support
  Multi-Node
  Multi-Pass
  Multi-Run
  Multi-Tile

Provide library of customizable recipes
Multiple results presentations
Document and share results

Multi-Node Analysis Recipes Library

CLI Usage: nsys recipe [<args>] <recipe name> [<recipe args>]

cuda_api_sum -- CUDA API Summary

cuda_api_sync -- CUDA Synchronization APIs

cuda_gpu_kern_pace -- CUDA GPU Kernel Pacing

cuda_gpu_kern_sum -- CUDA GPU Kernel Summary

cuda_gpu_mem_size_sum -- CUDA GPU MemOps Summary (by Size)

cuda_gpu_mem_time_sum -- CUDA GPU MemOps Summary (by Time)

cuda_gpu_time_util_map -- CUDA GPU Time Utilization Heatmap

cuda_memcpy_async -- CUDA Async Memcpy with Pageable Memory

cuda_memcpy_sync -- CUDA Synchronous Memcpy

cuda_memset_sync -- CUDA Synchronous Memset

diff -- Statistics Diff

dx12_mem_ops -- DX12 Memory Operations

gpu_gaps -- GPU Gaps

gpu_metric_util_map -- GPU Metric Utilization Heatmap

gpu_time_util -- GPU Time Utilization

mpi_gpu_time_util_map -- MPI and GPU Time Utilization Heatmap

mpi_sum -- MPI Summary

nccl_gpu_proj_sum -- NCCL GPU Projection Summary

nccl_sum -- NCCL Summary

nvtx_gpu_proj_pace -- NVTX GPU Projection Pacing

nvtx_gpu_proj_sum -- NVTX GPU Projection Summary

nvtx_gpu_proj_trace -- NVTX GPU Projection Trace

nvtx_pace -- NVTX Pacing

nvtx_sum -- NVTX Range Summary

osrt_sum -- OS Runtime Summary

and growing...

Connect Your AI Agent to Nsight Systems

Point your agent to the skill, then ask profiling questions in natural language.

- **Find the installed skill:** **`<nsight-systems-root>\ai-capabilities\SKILL.md`**
- **Reference the configuration:** For repeated use, reference **`SKILL.md`** from `AGENTS.md`, `CLAUDE.md`, or the equivalent agent configuration file.
- **Add this instruction:** *“For Nsight Systems tasks, follow* **`<nsight-systems-root>\ai-capabilities\SKILL.md`***.”*
- **One-time use:** For one-time use, include the **`SKILL.md`** path directly in your prompt.
- **Guided discovery:** **`SKILL.md`** guides the agent to the appropriate Nsight Systems documentation, analysis tools, and recipes.