



NVIDIA Developer Tools: Nsight Systems & Nsight Compute Intro

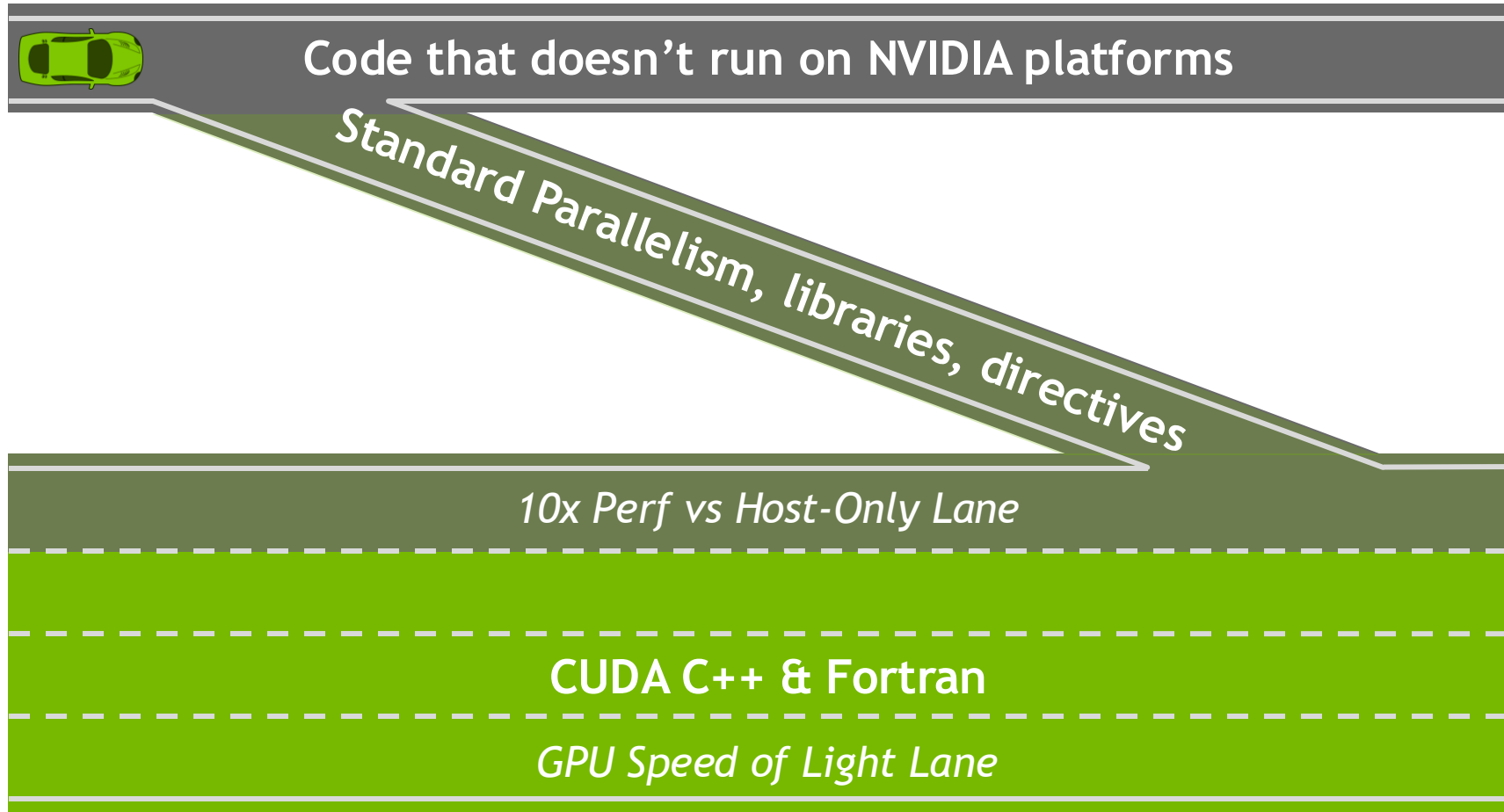
Outline for today

- Quick overview – Developer Tools Place in the Ecosystem
- NSight Systems, Nsight Compute
- Walkthrough of example:
 - Code available at https://github.com/ljdursi/alliance_oct24_nsight
 - Includes Nsight reports in case you're having trouble generating them
- Some advanced topics we didn't see
- Questions and Office Hours
- Let the Accelerator Working group know if there are other topics you'd like covered!



How we think about GPU development

GPU Computing Needs On-Ramps



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

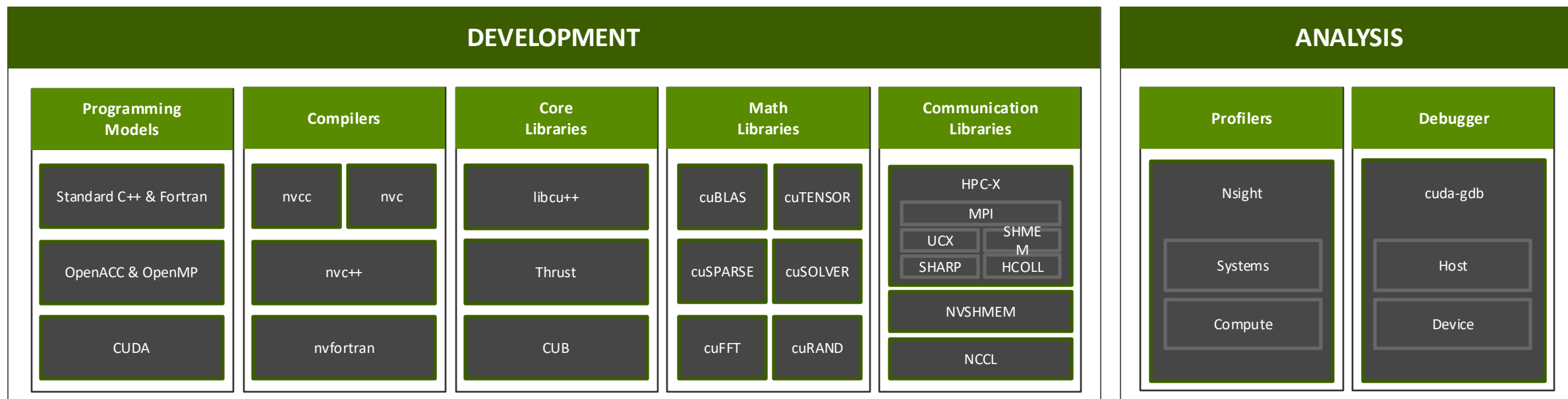
Data Analytics

AI

Quantum

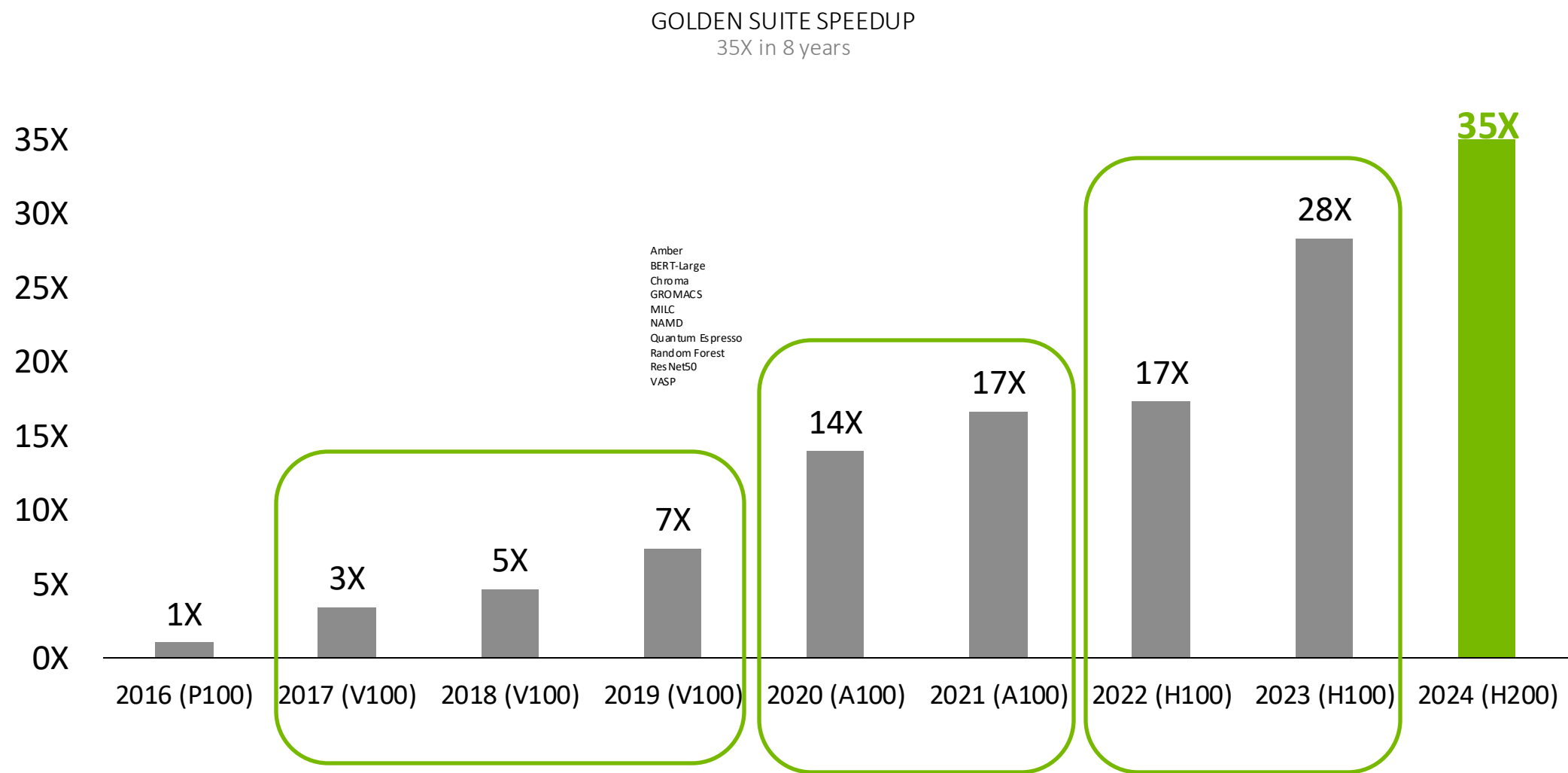
NVIDIA HPC SDK

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

Libraries, Tools Continually Improve Performance



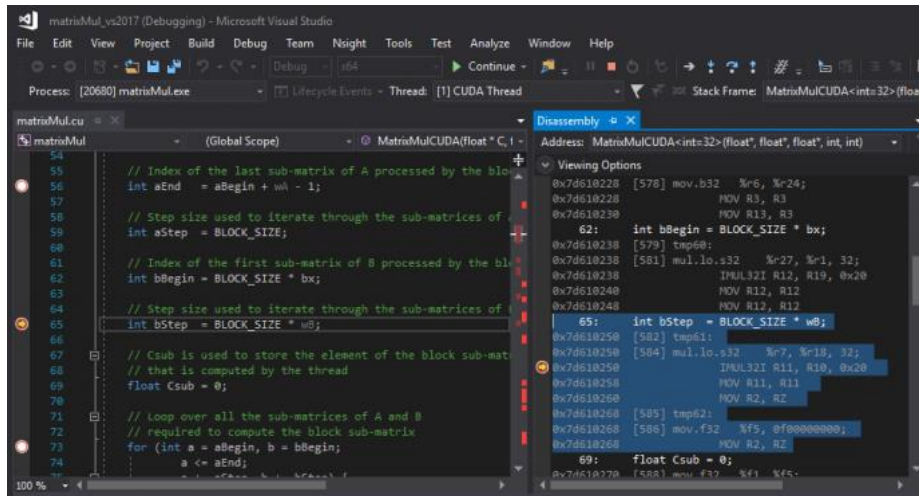
Geometric mean of application speedups vs. P100 in 2016 | benchmark applications | Amber [PME-Cellulose_NVE], Chroma [HMC], GROMACS [ADH Dodec], MILC [Apex Medium], NAMD [stmv_nve_cuda], PyTorch (BERT Large Fine Tuner), Quantum Espresso [AUSURF112-jR]; TensorFlow [ResNet-50], VASP 6 [Si Huge]; Random Forest make_blobs (160000 x 64 : 10).



Developer Tools Ecosystem

Developer tools

Debuggers: cuda-gdb, Nsight Visual Studio Edition



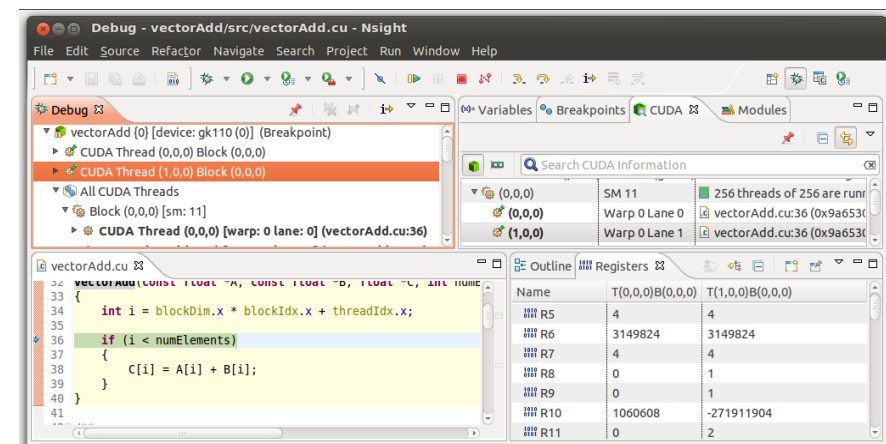
Profilers: Nsight Systems, Nsight Compute, NVIDIA Tools eXtension (NVTX)



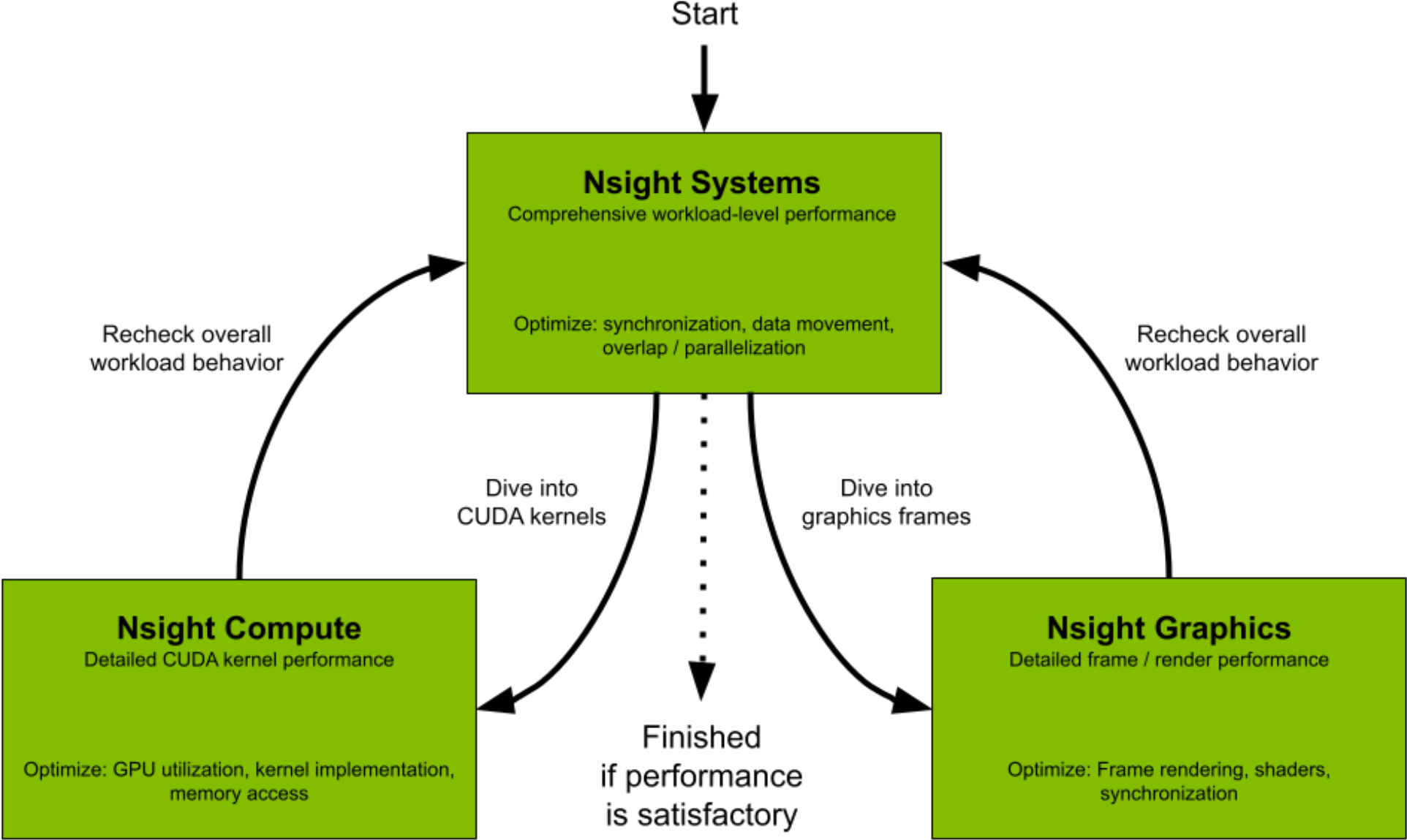
Correctness Checker: Compute Sanitizer

```
$ 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



Nsight Overview



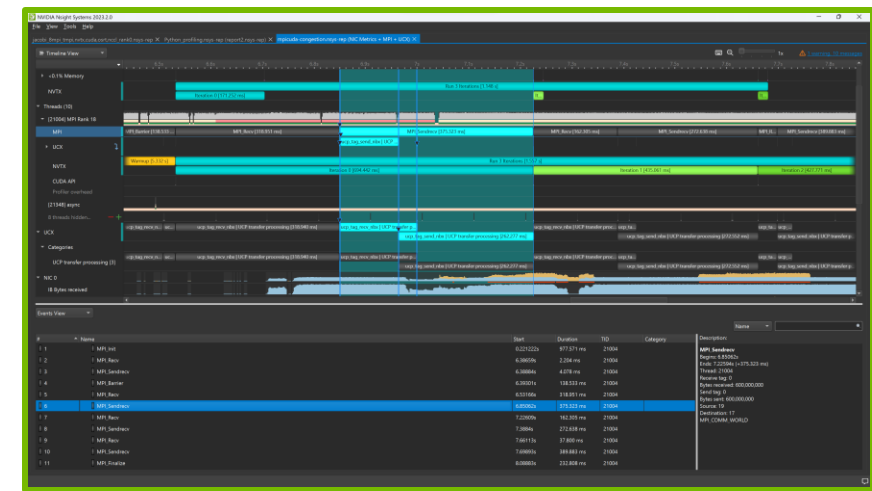
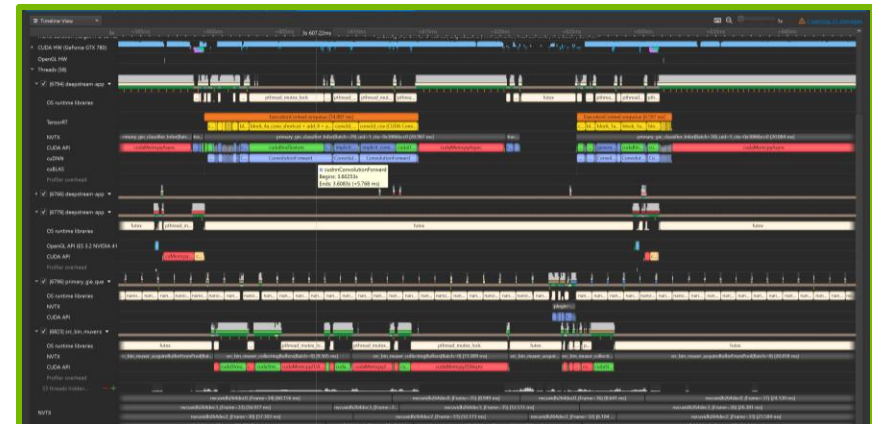


Nsight Systems

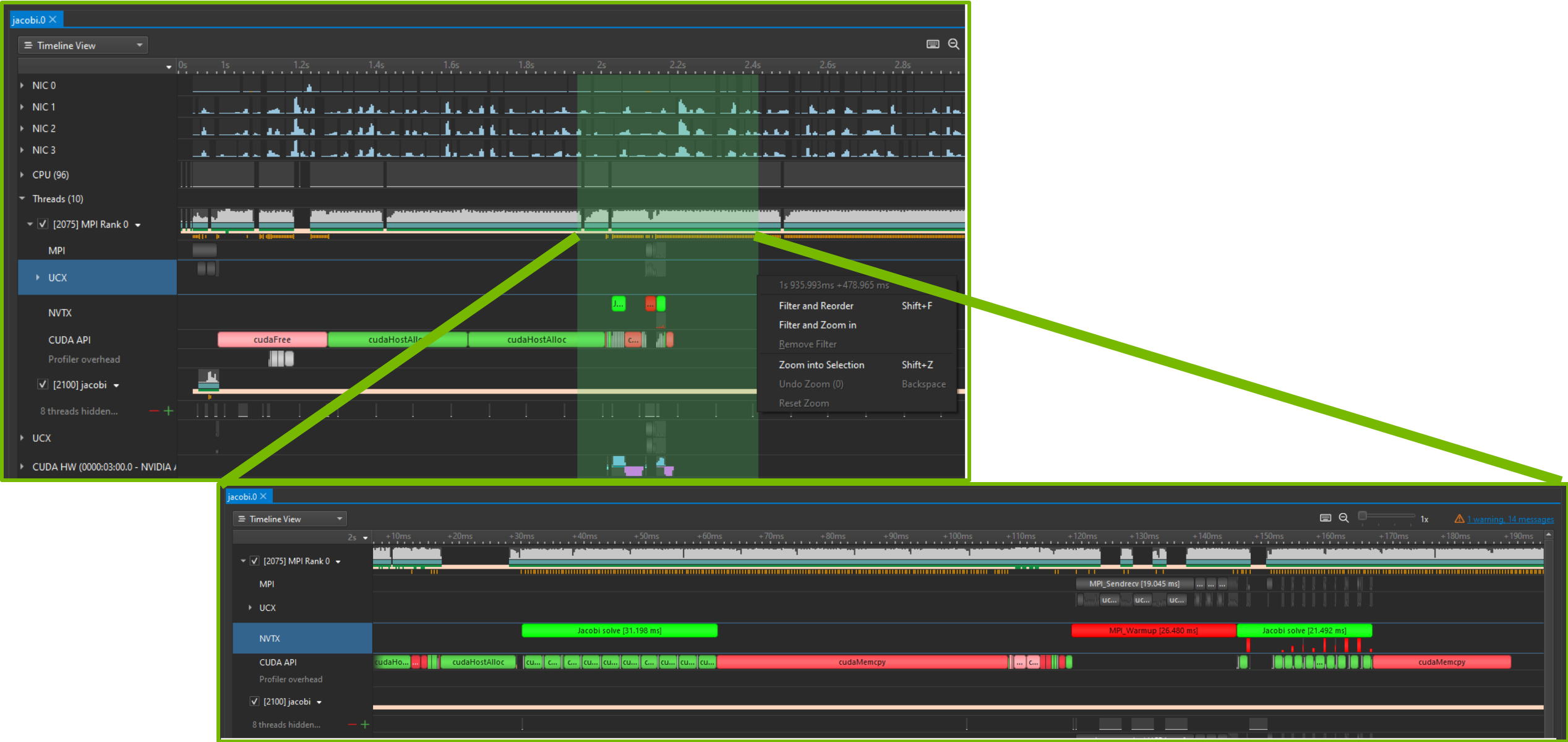
System Profiler

Key Features:

- System-wide application algorithm tuning
 - Multi-process tree support
- Locate optimization opportunities
 - Visualize millions of events on a very fast GUI timeline
 - Identify gaps of unused CPU and GPU time
- Balance your workload across multiple CPUs and GPUs
 - CPU algorithms, utilization and thread state
 - GPU streams, kernels, memory transfers, etc
- Command Line, Standalone, IDE Integration
- OS: Linux (x86, ARM, Tegra), Windows, macOS X (host)
- GPUs: Pascal+
- Docs/product: <https://developer.nvidia.com/nsight-systems>



Zoom/Filter to Exact Areas of Interest



NVIDIA Tools eXtension (NVTX)

- Decorate application source code with annotations (markers, ranges, nested ranges, ...) to help visualize execution with debugging, tracing and profiling tools

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

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

- Marker:

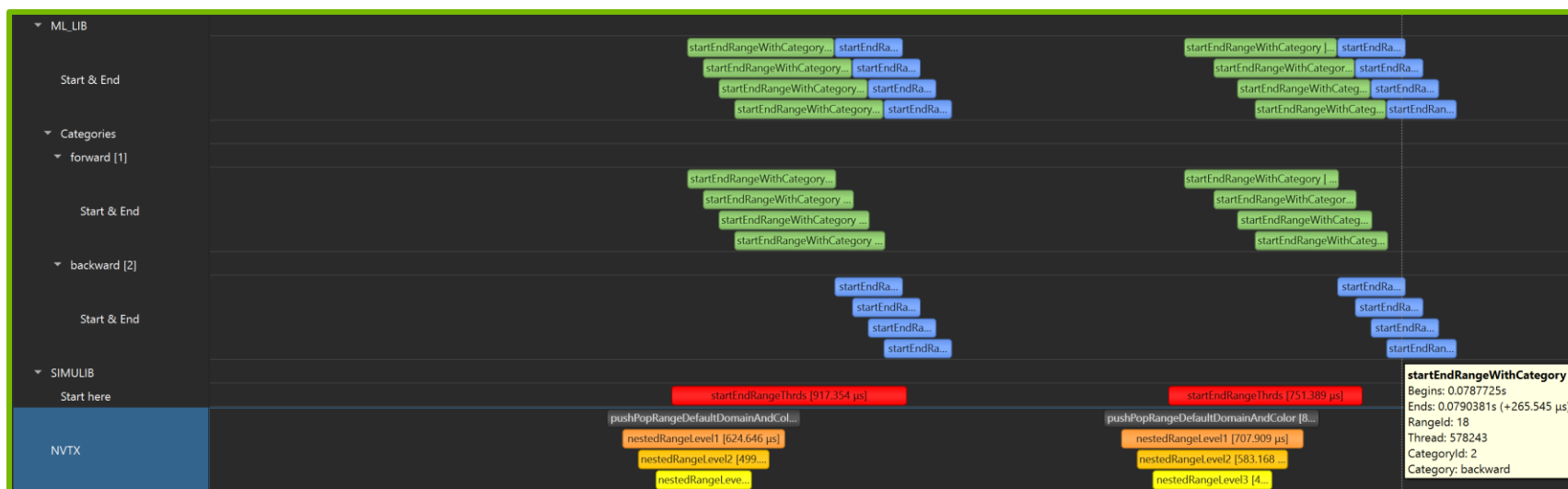
```
nvtxMark("This is a marker");
```

- Push-Pop range

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

- Start-End range

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



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

Python and NVTX

- Annotate Python code with NVTX

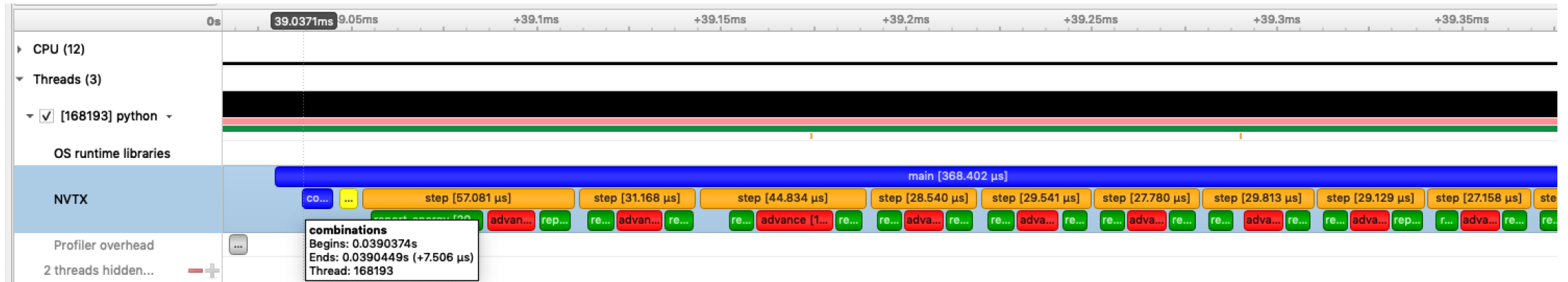
```
# demo.py

import time
import nvtx

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

my_function()
```

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



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
```
- Kernel's `perf_event_paranoid` value on the node has to be set to be 2 or less ([docs link](#)) or other processes (even those of the same user) can't access CPU performance information.



Nsight Compute

Kernel Profiler

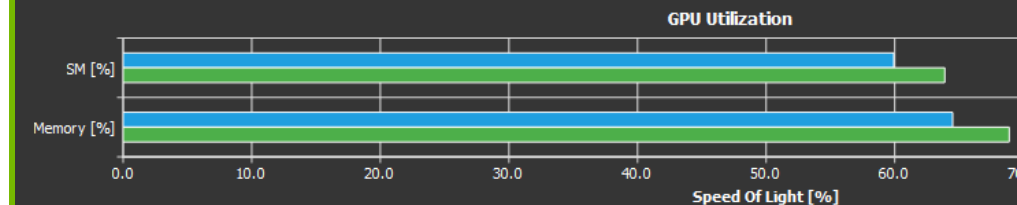
Key Features:

- Interactive CUDA API debugging and kernel profiling
- Built-in rules expertise
- Fully customizable data collection and display
- Command Line, Standalone, IDE Integration, Remote Targets
- OS: Linux (x86, Power, Tegra, Arm SBSA), Windows, macOS X (host only)
- GPUs: Volta+
- Docs/product: <https://developer.nvidia.com/nsight-compute>

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 percent of the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

SOL SM [%]	59.93	(-6.20%)	Duration [usecond]
SOL Memory [%]	64.50	(-6.38%)	Elapsed Cycles [cycle]
SOL L1/TEX Cache [%]	26.92	(-5.33%)	SM Active Cycles [cycle]
SOL L2 Cache [%]	64.50	(-6.38%)	SM Frequency [cycle/nsecond]
SOL DRAM [%]	51.55	(+84.34%)	DRAM Frequency [cycle/nsecond]



inst_executed [inst]	63,021,056 (284 instances)
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.sum	0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.sum	0
l1tex__data_bank_reads.avg.pct_of_peak_sustained_elapsed [%]	9.66
l1tex__data_bank_writes.avg.pct_of_peak_sustained_elapsed [%]	3.23
l1tex__data_pipe_lsu_wavefronts.avg.pct_of_peak_sustained_elapsed [%]	46.16
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_read.sum	25,165,824
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_read.sum.pct_of_peak_sustained_active [%]	40.75
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_write.sum	2,097,152
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_write.sum.pct_of_peak_sustained_active [%]	3.40
l1tex__data_pipe_tex_wavefronts.avg.pct_of_peak_sustained_elapsed [%]	0
l1tex__f_wavefronts.avg.pct_of_peak_sustained_elapsed [%]	0.00
l1tex__lsu_writeback_active.avg.pct_of_peak_sustained_elapsed [%]	42.59
l1tex__lsu_writeback_active.sum [cycle]	27,803,648
l1tex__lsu_writeback_active.sum.pct_of_peak_sustained_active [%]	45.03
l1tex__lsuin_requests.avg.pct_of_peak_sustained_elapsed [%]	66.00
l1tex__m_l1tex2xbar_req_cycles_active.avg.pct_of_peak_sustained_elapsed [%]	3.40
l1tex__m_l1tex2xbar_write_bytes.sum [Mbyte]	4.19
l1tex__m_l1tex2xbar_write_bytes_mem_global_op_red.sum [byte]	0

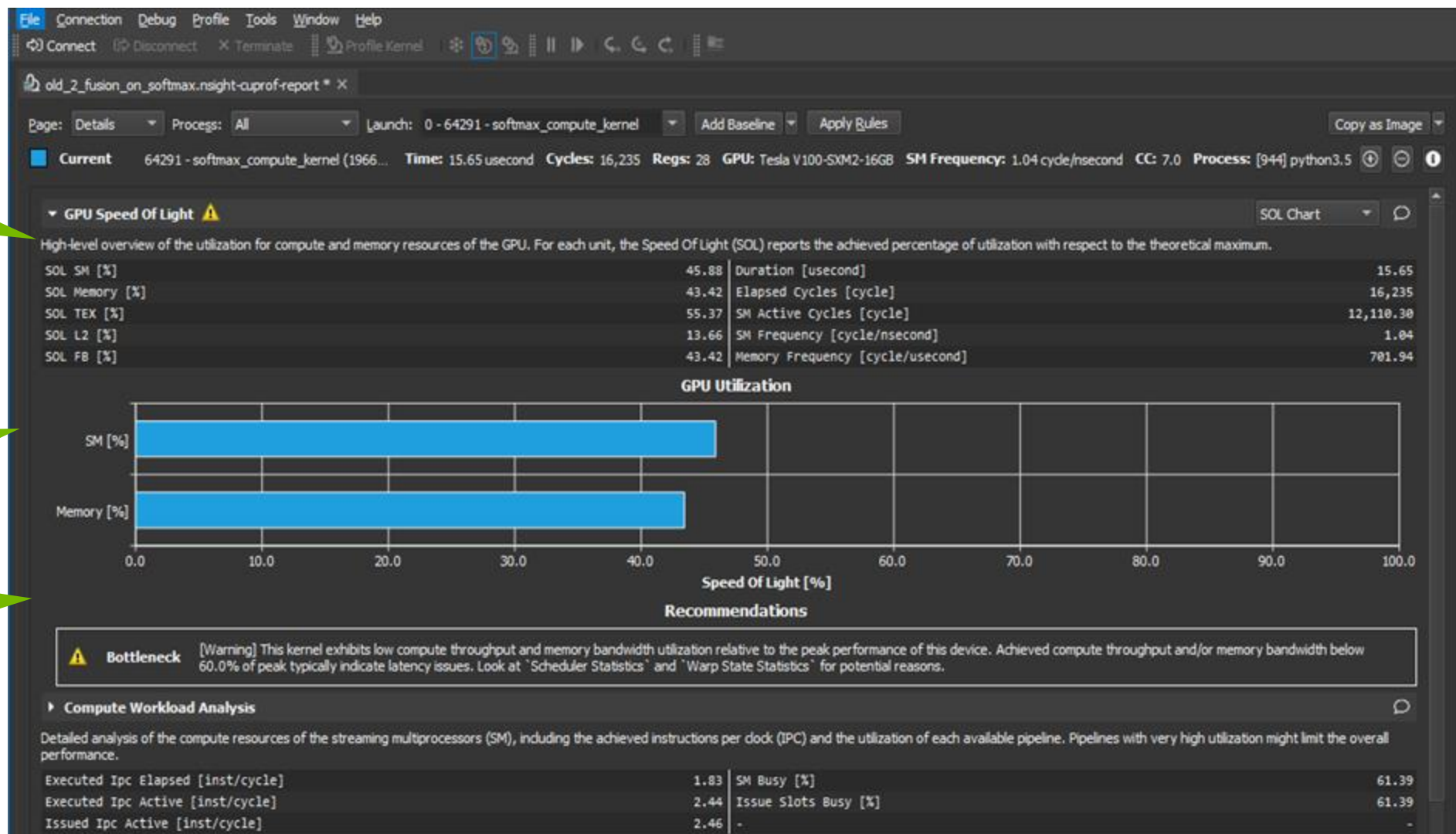
@P0 EXIT	6	108	49	1,404,672
IADD3 R7, P2, R0, UR7, RZ	7	177	95	1,401,344
IADD3 R6, P1, R4, UR4, RZ	8	30	9	1,401,344
ISETP.GE.U32.AND P0, PT, R7, UR5, PT	8	1	1	1,401,344
IADD3.X R8, R2, UR8, RZ, P2, !PT	8	1	0	1,401,344
IMAD.X R7, RZ, RZ, R5, P1	9	1	0	1,401,344
ISETP.GE.U32.AND.EX P0, PT, R8, UR6, PT, P0	9	106	35	1,401,344
STG.E.U8 [R6.64], R3	8	116	75	1,401,344
@P0 EXIT	8	92	33	1,401,344
IADD3 R8, P2, R0, UR9, RZ	9	45	14	1,397,120
IADD3 R6, P1, R6, UR4, RZ	9	248	145	1,397,120
ISETP.GE.U32.AND P0, PT, R8, UR5, PT	9	57	17	1,397,120
IADD3.X R8, R2, UR12, RZ, P2, !PT	9	1	1	1,397,120
IMAD.X R7, RZ, RZ, R7, P1	9	7	0	1,397,120
ISETP.GE.U32.AND.EX P0, PT, R8, UR6, PT, P0	9	94	15	1,397,120
STG.E.U8 [R6.64], R3	8	104	61	1,397,120

Nsight Compute GUI Interface

Targeted metric sections

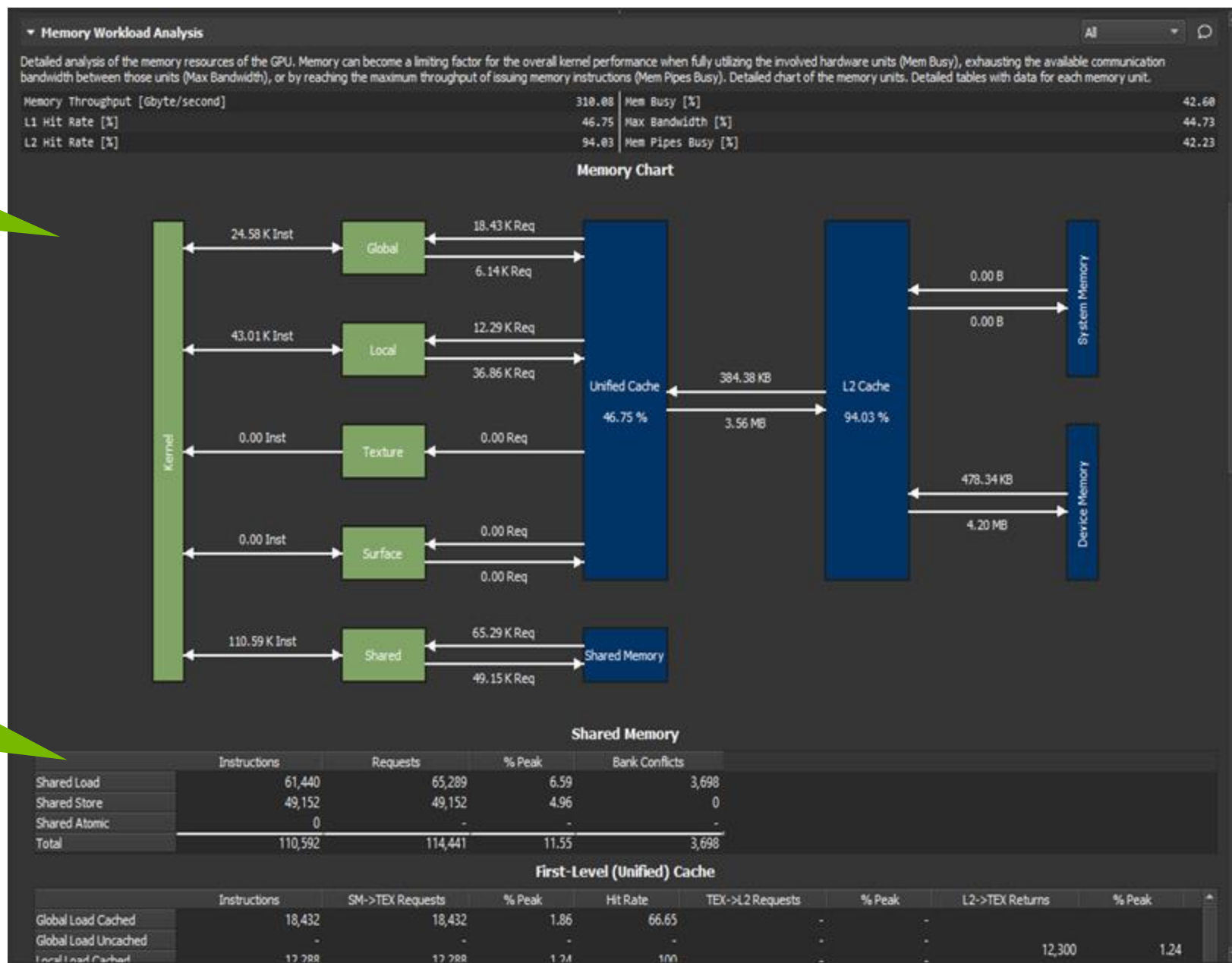
Customizable data collection and presentation

Built-in expertise for Guided Analysis and optimization

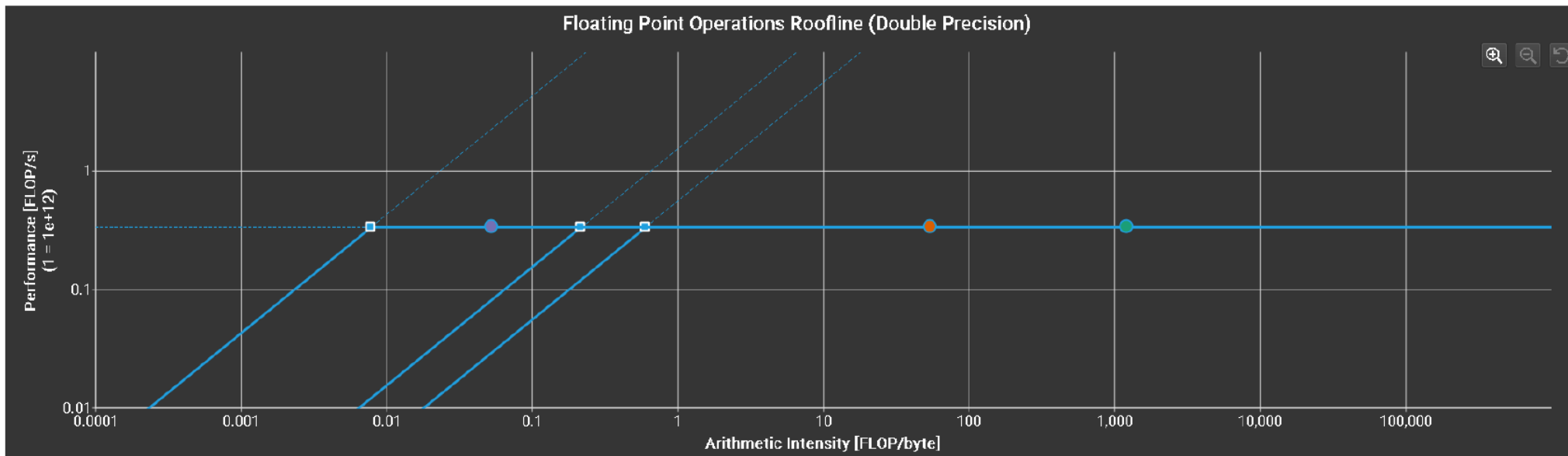


Visual memory analysis chart

Metrics for peak performance ratios



Hierarchical Roofline



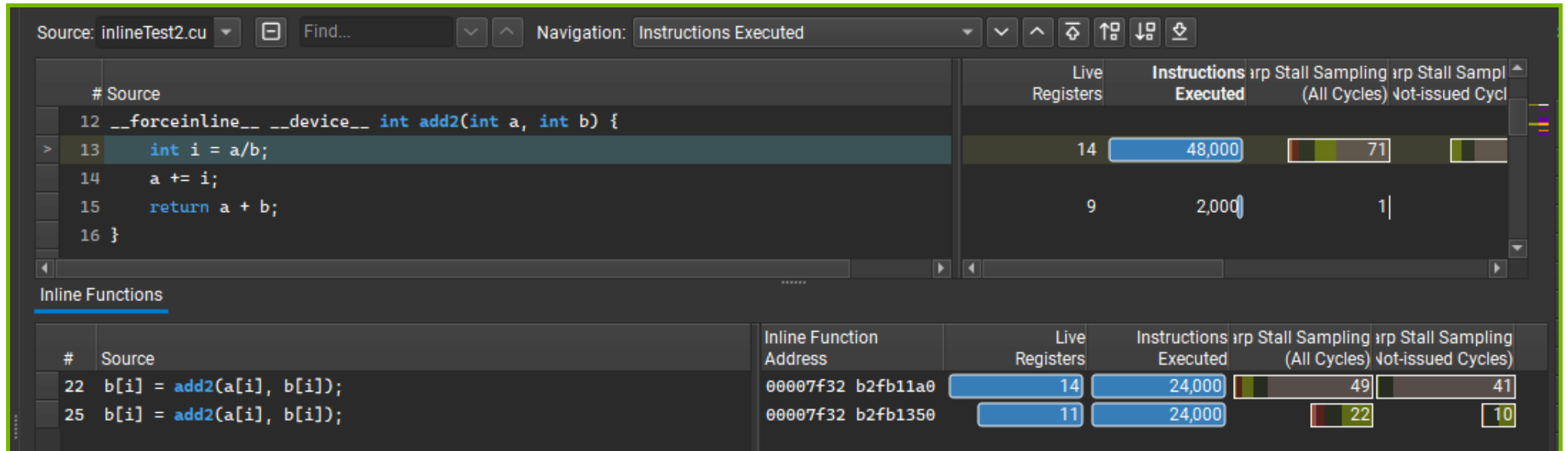
- Visualize multiple levels of the memory hierarchy
- Identify bottlenecks caused by memory limitations
- Determine how modifying algorithms may (or may not) impact performance

Sections/Rules Info			
Sections/Rules			
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...

Inline Function Table

Shipped with Nsight Compute 2023.1

- Metrics can be analyzed per inline site or aggregated for the entire function
- Use compiler `--lineinfo` flag to generate symbols
- Identify specific underperforming calls and outliers



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 will profile everything and take a long time)
- `ncu` needs access to the GPU performance counters, which may require a parameter change to the `nvidia kernel module` (`options nvidia NVreg_RestrictProfilingToAdminUsers=0`)

The background features a series of diagonal, overlapping green bands that create a sense of depth and movement. A solid green vertical bar is positioned on the far left side of the frame.

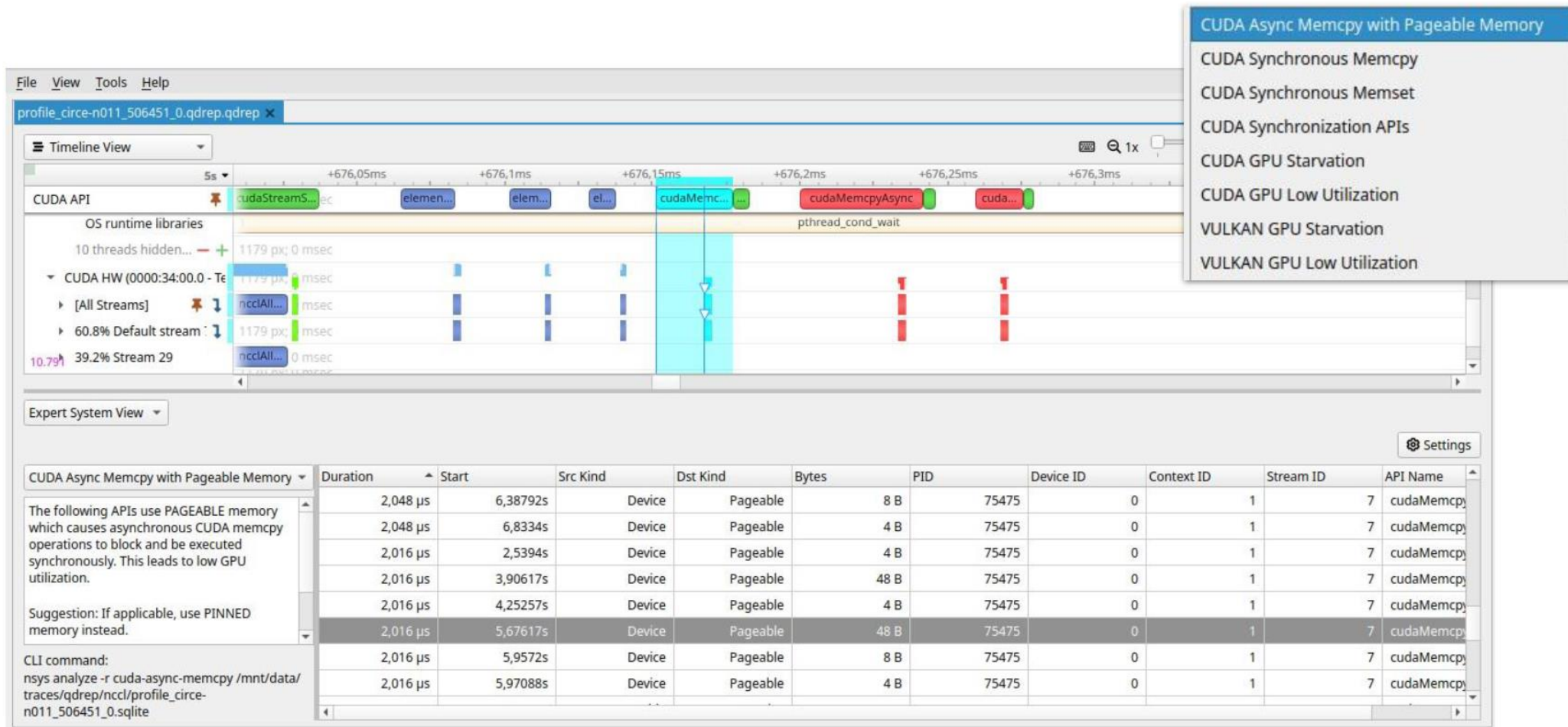
Let's Get Profiling!



**Some Advanced Features We Didn't
See Today**

Expert Systems & Statistics

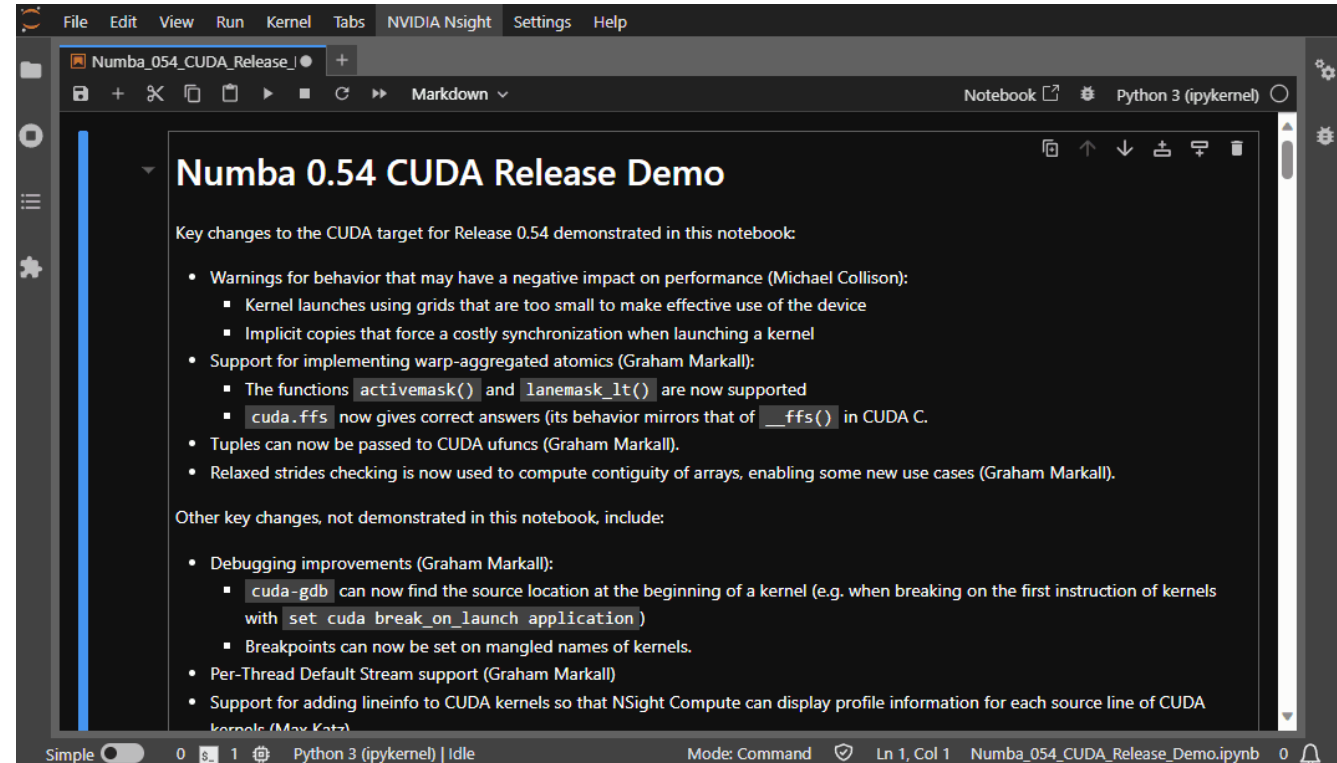
Built-in data analytics with advice





JupyterLab Integration Updates

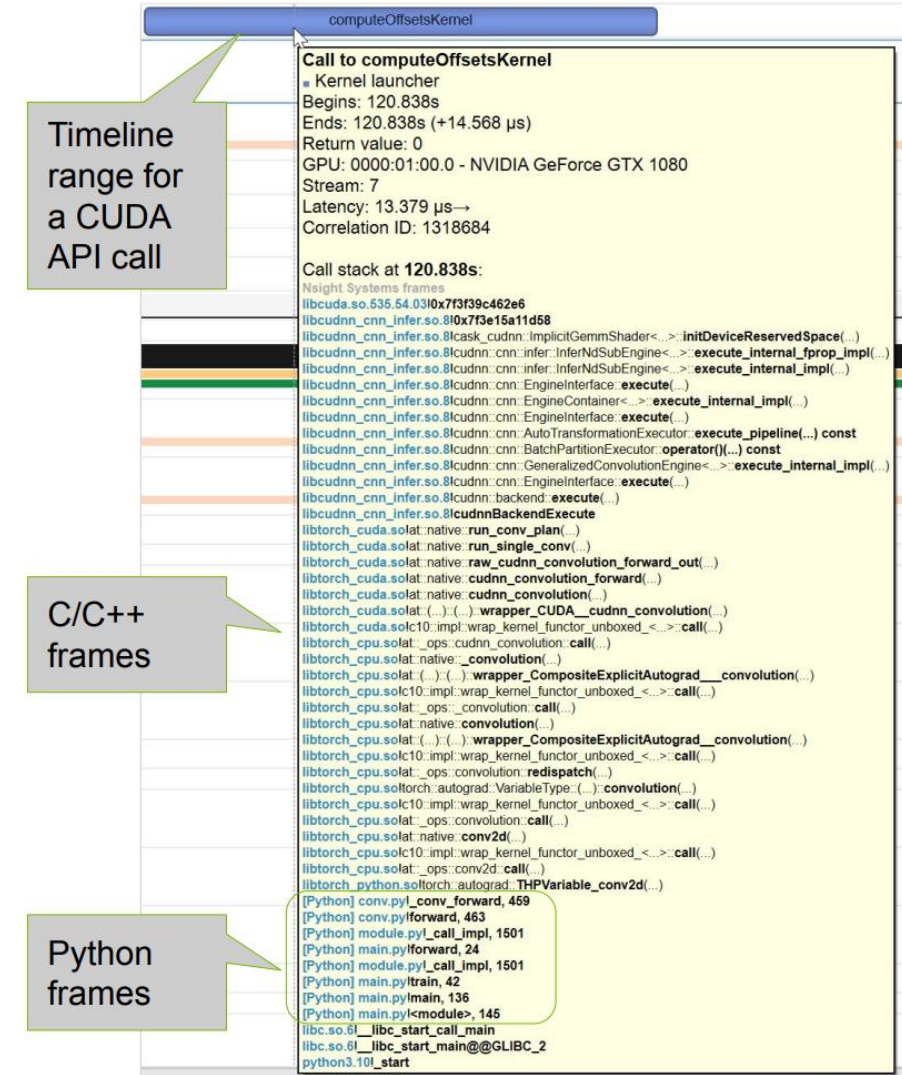
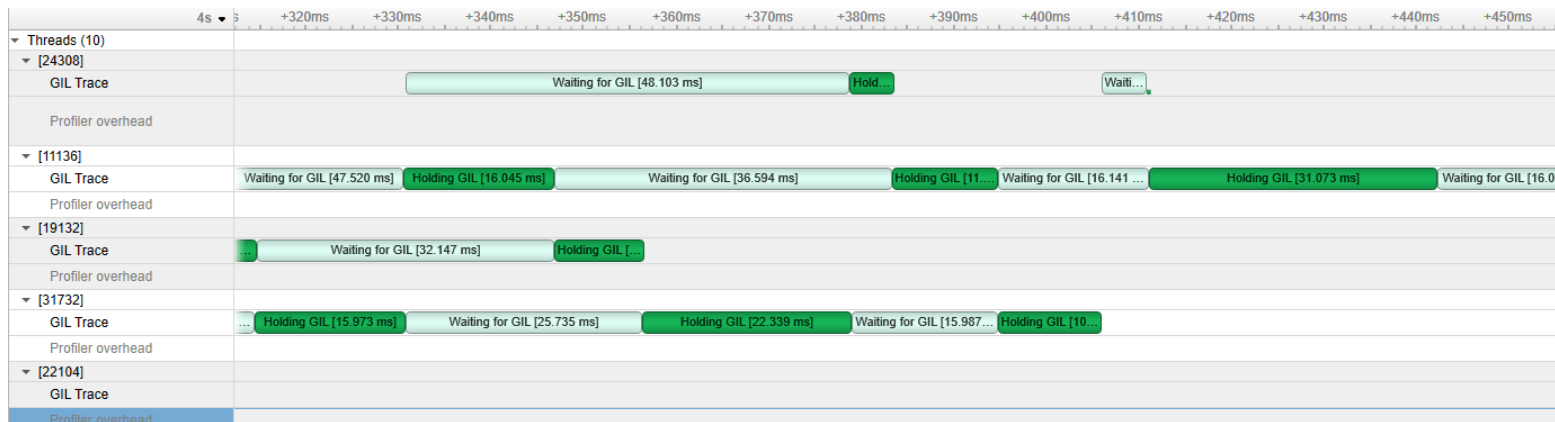
- Extension to JupyterLab
- Profile individual Jupyter cells
- Text-based results can be viewed directly in Jupyter
- Launch **new** remote GUI streaming container directly in JupyterLab
 - Servers without X, Windowing Manager, ...
 - Container with X, WM, & WebRTC server
 - Dockerfile inside Nsight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





Python Profiling Updates

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

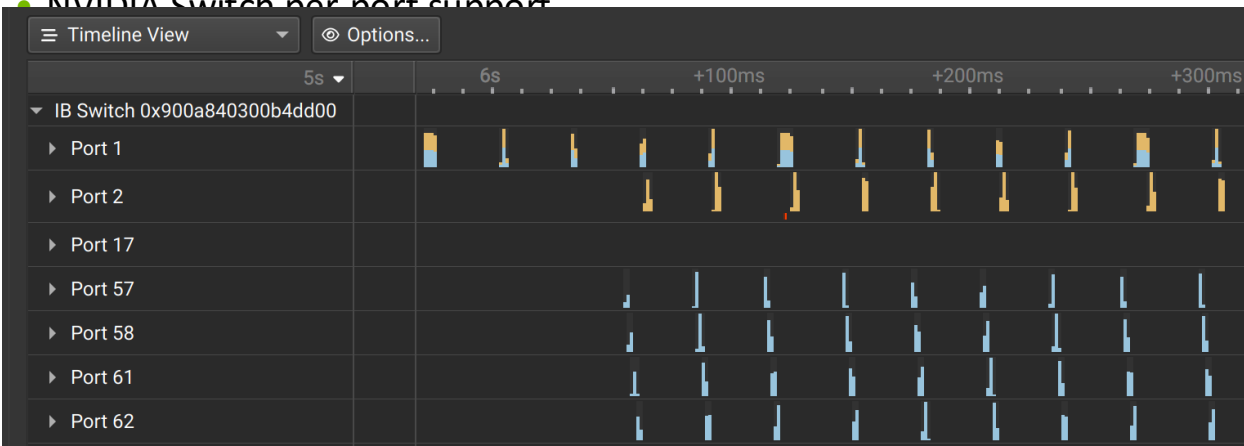




Cluster and Recipe Framework Improvements

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

• NVIDIA Switch port support



```
workstation: /develop/Archive/CSP/devtools-sidecar-injector$ kubectl get pods -A
```

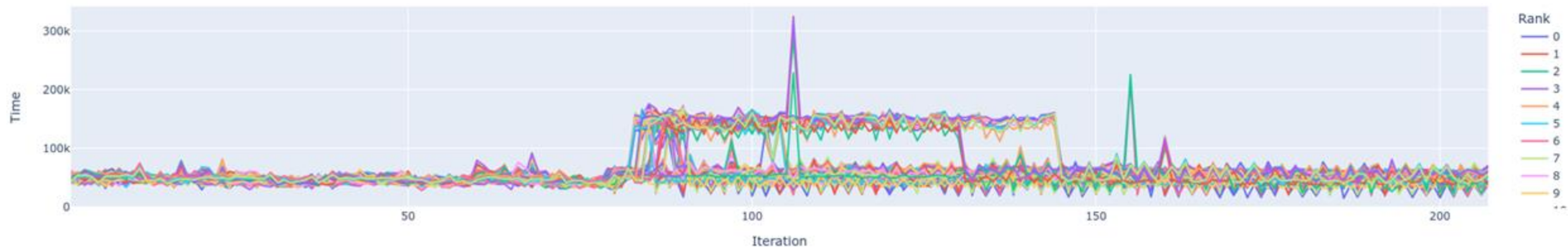
NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
example-ns	cuda-vector-add-69c5cb6b7c-r542t	1/1	Running	0	34s
gmp-system	alertmanager-0	2/2	Running	0	3d16h
gmp-system	collector-sd8ln	2/2	Running	0	3d16h
gmp-system	collector-tsjd7	2/2	Running	0	46m
gmp-system	gmp-operator-69f4b6cb87-1xfk5	1/1	Running	0	46h
gmp-system	rule-evaluator-9bd9c559f-2kzkh	2/2	Running	2 (3d16h ago)	3d16h
gpu-operator	gpu-feature-discovery-lwd45	1/1	Running	0	45m
gpu-operator	gpu-operator-999cc8dcc-cj5hc	1/1	Running	10 (46h ago)	3d15h
gpu-operator	gpu-operator-node-feature-discovery-gc-7cc7ccfff8-2cgbh	1/1	Running	0	3d15h
gpu-operator	gpu-operator-node-feature-discovery-master-d8597d549-1t7vj	1/1	Running	0	3d15h
gpu-operator	gpu-operator-node-feature-discovery-worker-hcmr7	1/1	Running	0	46m
gpu-operator	gpu-operator-node-feature-discovery-worker-rvvcz	1/1	Running	9 (46h ago)	3d15h
gpu-operator	nvidia-container-toolkit-daemonset-lqgv7	1/1	Running	0	45m
gpu-operator	nvidia-cuda-validator-k6bph	0/1	Completed	0	41m
gpu-operator	nvidia-dcgm-exporter-29kbz	1/1	Running	0	45m
gpu-operator	nvidia-device-plugin-daemonset-n7rjl	1/1	Running	0	45m
gpu-operator	nvidia-driver-daemonset-jb4dw	1/1	Running	0	45m
gpu-operator	nvidia-operator-validator-56nc4	1/1	Running	0	45m
kube-system	event-exporter-gke-754cff8686-mv585	2/2	Running	0	3d16h
kube-system	fluentbit-gke-d8kg2	2/2	Running	0	46m
kube-system	fluentbit-gke-lrjvb	2/2	Running	0	3d16h
kube-system	gke-metadata-server-8qm55	1/1	Running	0	46m
kube-system	gke-metadata-server-kfcj8	1/1	Running	0	3d16h
kube-system	gke-metrics-agent-nrgcn	2/2	Running	0	46h
kube-system	gke-metrics-agent-x8rcr	2/2	Running	0	46m
kube-system	konnectivity-agent-7f8fc89f85-c96jx	2/2	Running	0	3d15h
kube-system	konnectivity-agent-7f8fc89f85-stgls	2/2	Running	0	3d16h
kube-system	konnectivity-agent-autoscaler-8fff668b4-rlz7q	1/1	Running	0	3d16h
kube-system	kube-dns-577947fcfc-2g4x4	4/4	Running	0	3d15h
kube-system	kube-dns-577947fcfc-hrsdh	4/4	Running	0	3d16h
kube-system	kube-dns-autoscaler-755c7d9df5-9b5bv	1/1	Running	0	3d16h
kube-system	kube-proxy-gke-nsight-load-test-tf-nsight-load-t-03e52019-1z5m	1/1	Running	0	3d16h
kube-system	kube-proxy-gke-nsight-load-test-tf-nsight-load-t-e8d740d6-2jbbh	1/1	Running	0	46m
kube-system	l7-default-backend-9b4f84c76-wlwnl	1/1	Running	0	3d16h
kube-system	metrics-server-v0.6.3-b76d4c5f8-qvchh	2/2	Running	0	3d16h
kube-system	netd-mnsj9	1/1	Running	0	46m
kube-system	netd-vbc92	1/1	Running	0	3d16h
kube-system	nvidia-gpu-device-plugin-small-ubuntu-54pl6	0/1	Init:0/2	0	46m
kube-system	pdcsi-node-mm2s4	2/2	Running	0	46m
kube-system	pdcsi-node-w8mft	2/2	Running	0	3d16h
nvidia-devtools-sidecar-injector	nvidia-devtools-sidecar-injector-676b496845-jt9rc	1/1	Running	0	46s

```
workstation: /develop/Archive/CSP/devtools-sidecar-injector$ # Using nsys_k8s.py we can control the profiling of the containers.
./nsys_k8s.py nsys stop
Executing command: /mnt/nv/bin/nsight-systems/target-linux-x64/nsys stop --session k8s_auto_56b82acc
Output from pod cuda-vector-add-69c5cb6b7c-r542t, container cuda-vector-add:
Generating '/tmp/nsys-report-f50f.qdstrm'
[1/1] [=====100%] auto_sleep_example-ns_cuda-vector-add-69c5cb6b7c__cuda-vector-add-1708078828373_56b82acc.nsys-rep
Generated:
/home/auto_sleep_example-ns_cuda-vector-add-69c5cb6b7c__cuda-vector-add-1708078828373_56b82acc.nsys-rep
-workstation: /develop/Archive/CSP/devtools-sidecar-injector$
```

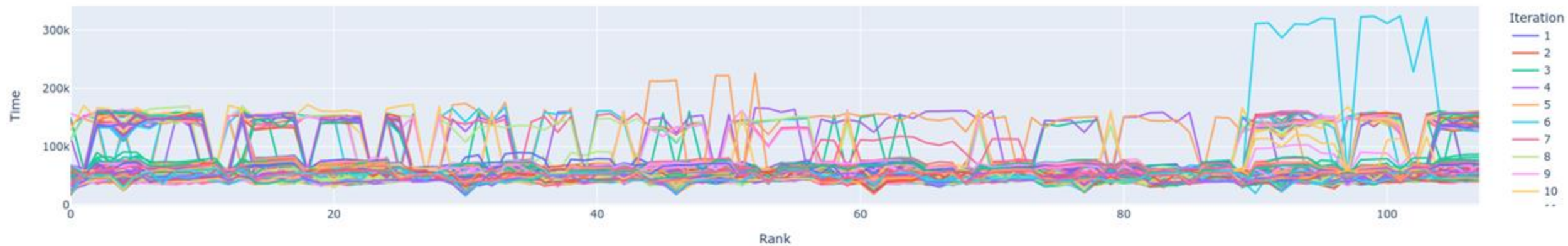
MULTI-REPORT ANALYSIS

Ex: Jupyter notebook output of NCCL barrier time across 128 GPUs
<https://www.nvidia.com/en-us/on-demand/session/gtcspring23-s51421/>

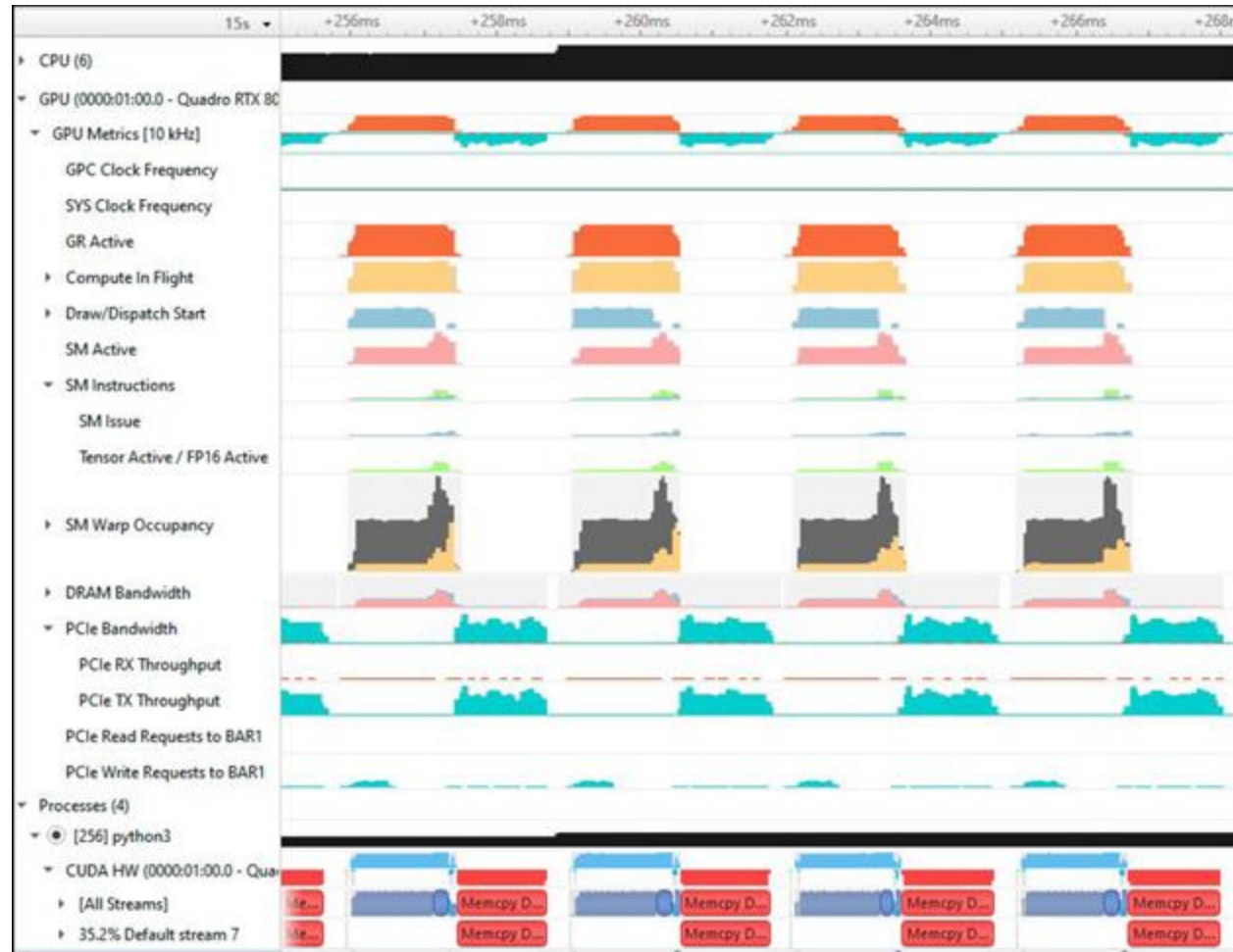
Progress - Iterations defined by duration of ncclKernel_AllReduce_TREE_LL_Sum_int64_t



Consistency - Iterations defined by duration of ncclKernel_AllReduce_TREE_LL_Sum_int64_t



GPU Metrics Sampling



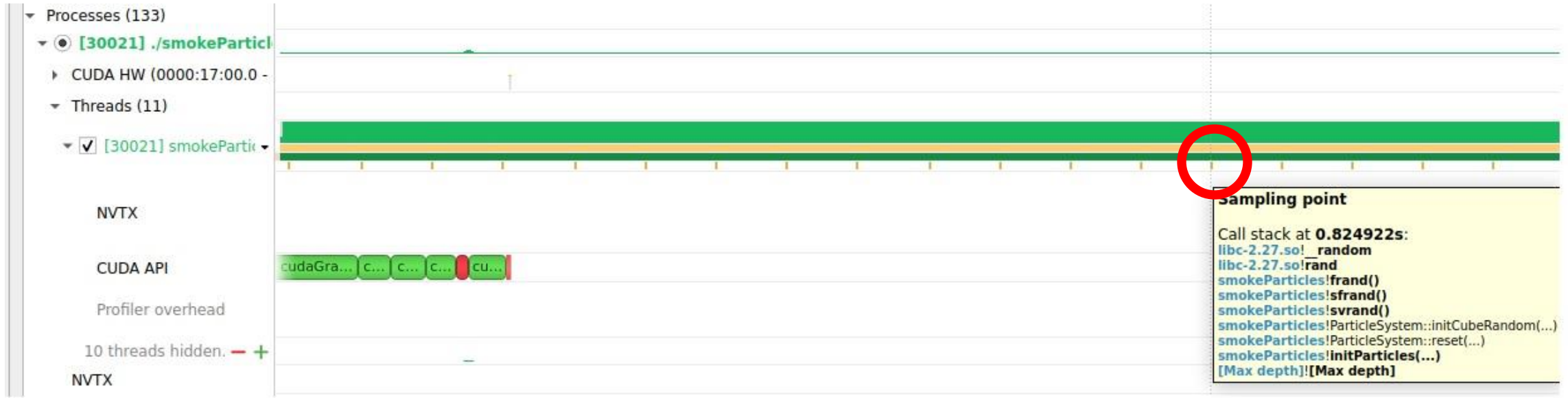
- Useful GPU utilization metrics, but no kernel names / correlation

Interpreting GPU Sampling Metrics

- GR Activity -> GPU is doing work
 - SM, NVENC, NVDEC, Graphics
- SM Activity -> Utilizing width of GPU
 - If low, modify kernel grid dimension or increase batch size
- SM Instruction Issued -> GPU is performing lots of instructions
 - Stalled waiting on memory?
 - Not enough warps to cover memory latency? Issue larger kernel block dimensions.
- SM Instructions tensor activity -> Tensor core utilization
 - Performance up, SM instructions can drop (depending on arch)
 - Can be limited by shared memory, waiting for loads
- Note: Requires disabling DCGM and DL built-in profilers

CPU IP/Backtrace Sampling

Placing a mouse cursor on a specific sample (orange-yellow tick mark) causes backtrace tooltip to be displayed

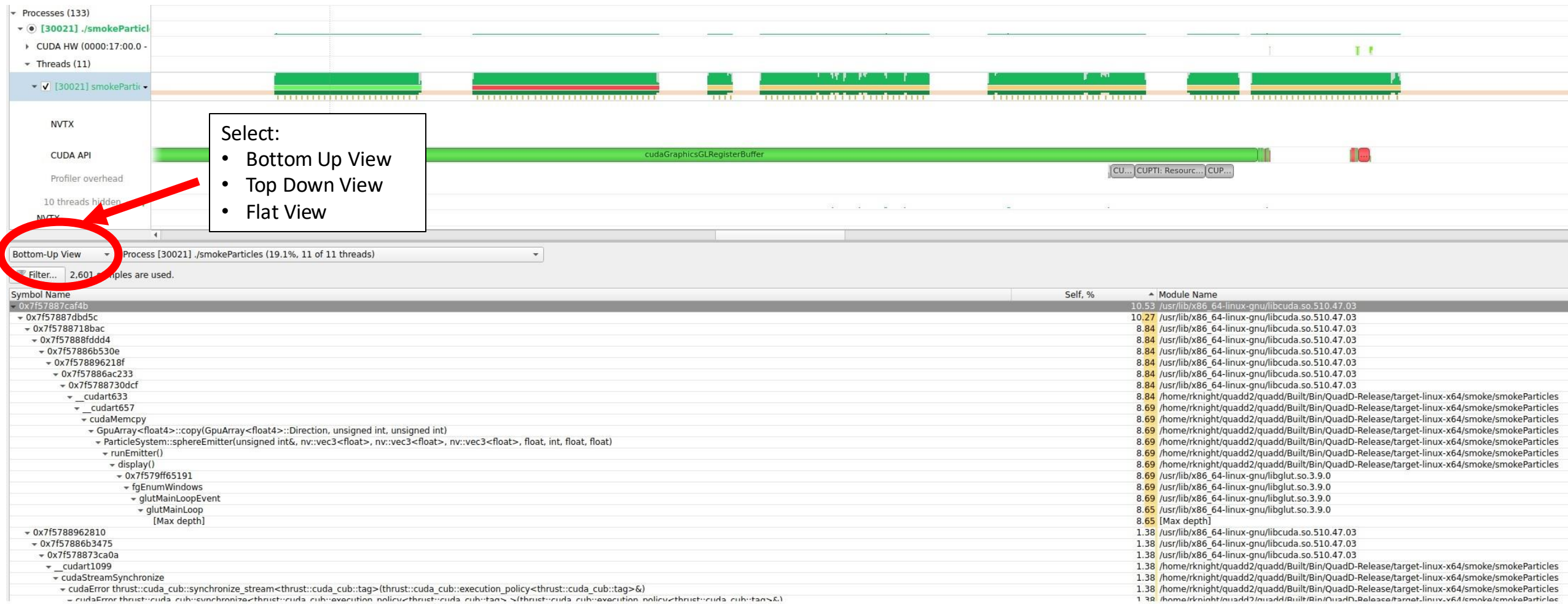


Stack depth is defined by callstack mechanism selected

See nsys CLI `--backtrace` switch

- Intel Last Branch Register (fast, limited depth)
- Frame Pointers (fastest, probably requires recompile)
- DWARF (best depth, most overhead – default on ARM systems)
 - Also, see nsys CLI `--samples-per-backtrace` switch

CPU IP/Backtrace Sampling Summary / Histogram



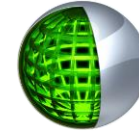
The summary enables CPU hotspot identification by identifying hot paths.

Top-Down view:

The **Self** column shows the percentage of IP/backtrace samples that were collected while a specific function was executing.

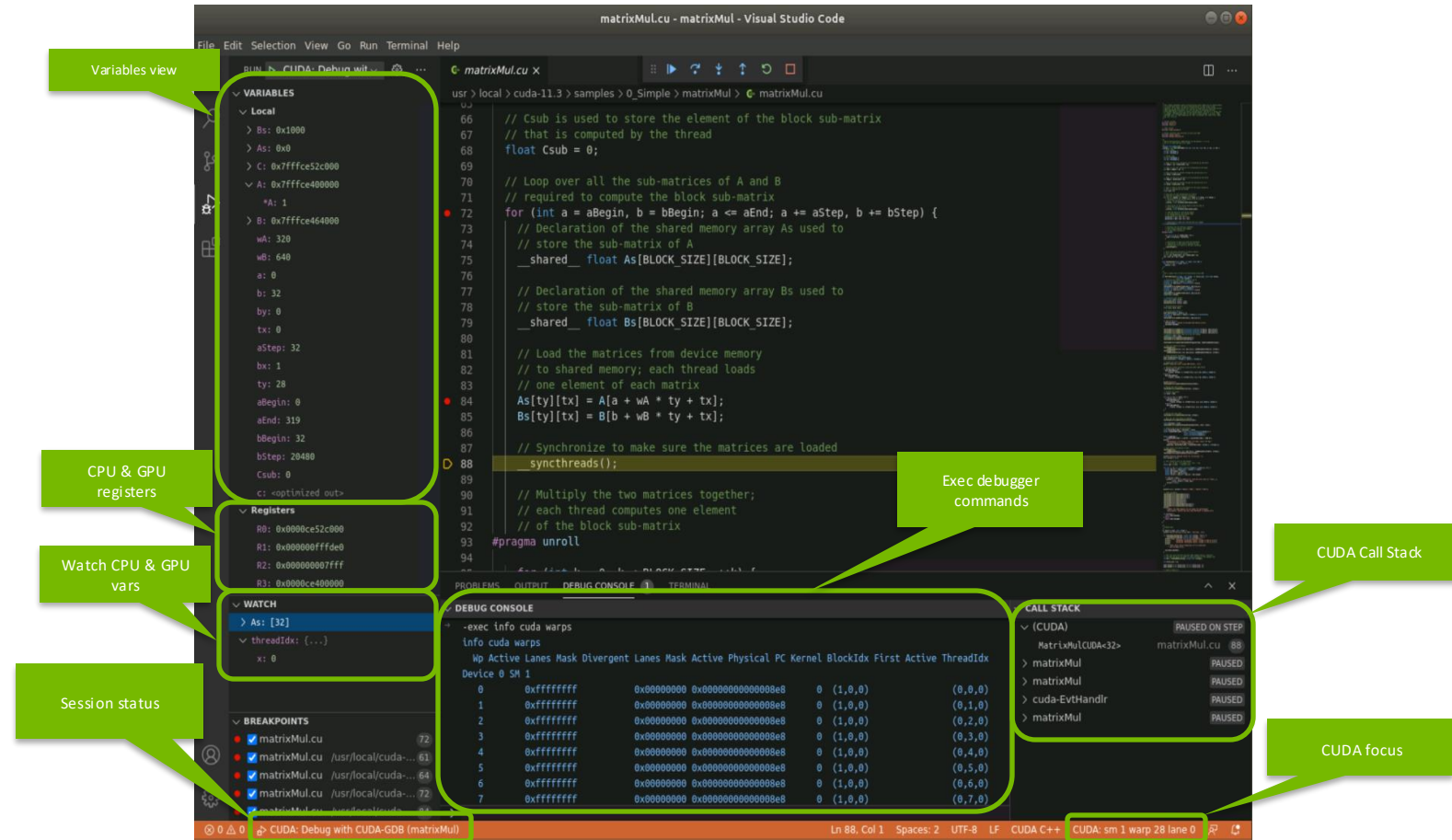
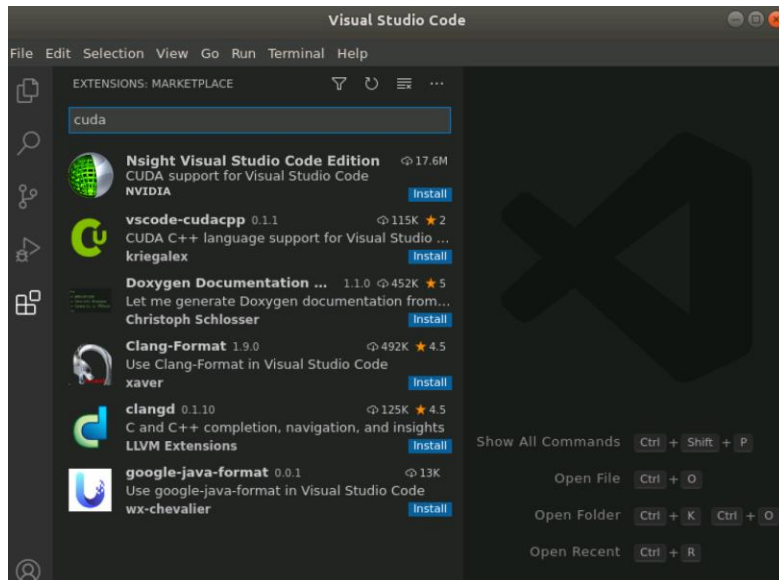
The **Total** column shows the percentage of IP/backtrace samples that were collected while that function and all of its children functions were executing.

Nsight Visual Studio Code Edition



Visual Studio Code extensions that provides:

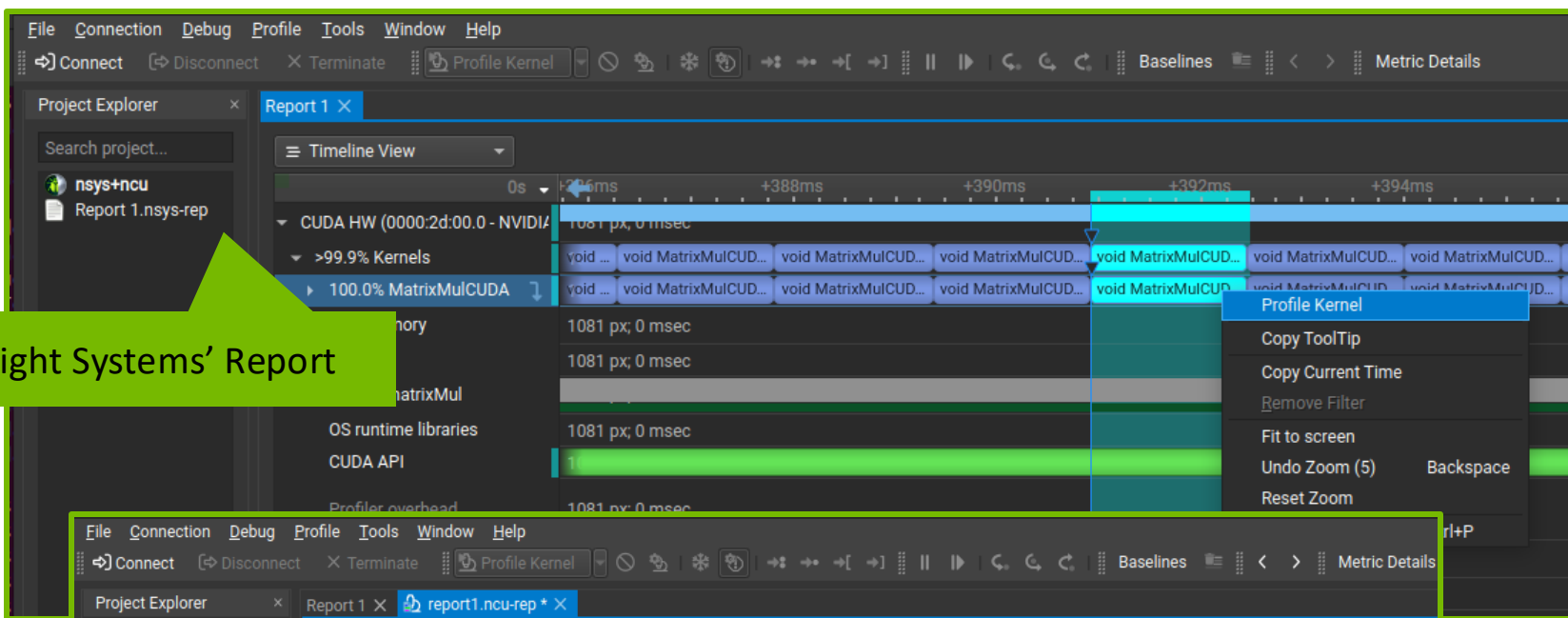
- CUDA code syntax highlighting
- CUDA code completion
- Build warning/errors
- Debug CPU & GPU code
- Remote connection support via SSH
- Available on the VS Code Marketplace now!



<https://developer.nvidia.com/nsight-visual-studio-code-edition>

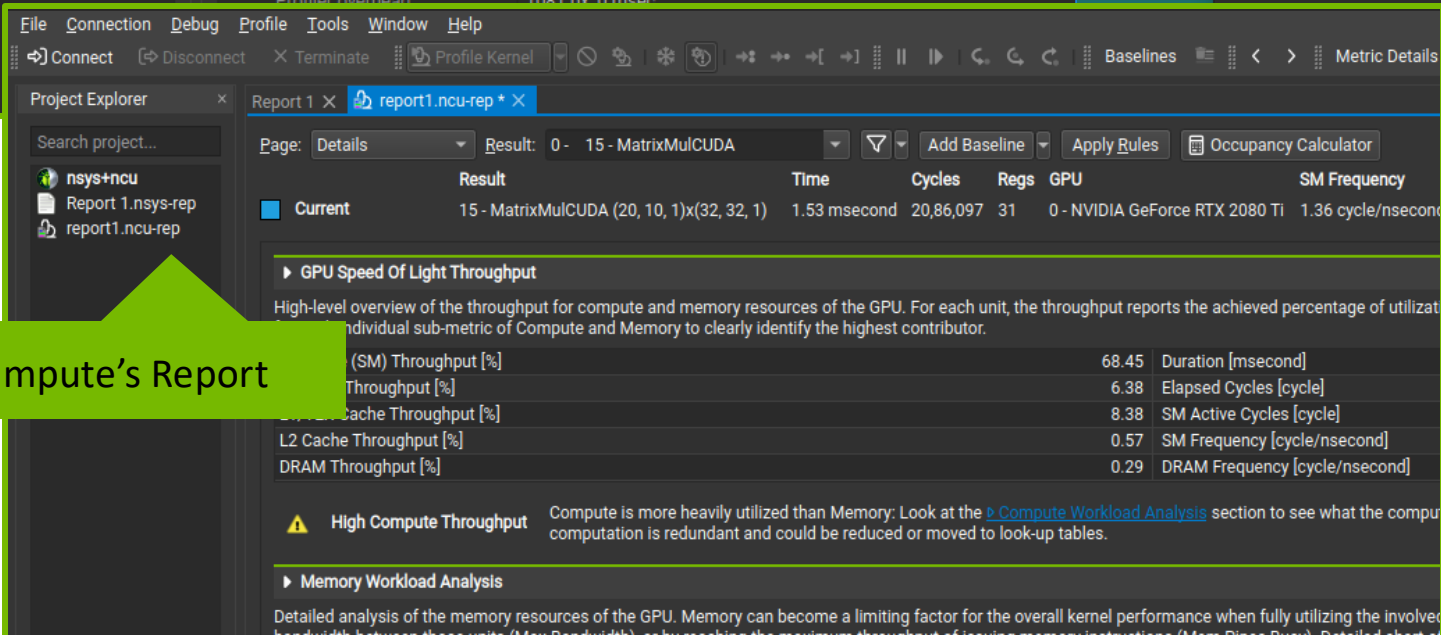
Integrating NSIGHT SYSTEMS basic trace INTO nsight COMPUTE

- Use “System Trace” activity in the connection dialog
- Identify long kernels or compute-bound bottlenecks
- Right-click kernel in timeline to quickly launch profile
- Nsight Compute automatically filters to selected kernel



Nsight Systems' Report

The screenshot shows the Nsight Systems interface with a timeline view. A context menu is open over a kernel, showing options like 'Profile Kernel', 'Copy ToolTip', 'Copy Current Time', 'Remove Filter', 'Fit to screen', 'Undo Zoom (5)', and 'Reset Zoom'.



Nsight Compute's Report

The screenshot shows the Nsight Compute interface with a detailed report for a specific kernel. The report includes a table of performance metrics and a summary of the GPU speed of light throughput.

	Result	Time	Cycles	Regs	GPU	SM Frequency
Current	15 - MatrixMulCUDA (20, 10, 1)x(32, 32, 1)	1.53 msecond	20,86,097	31	0 - NVIDIA GeForce RTX 2080 Ti	1.36 cycle/nsecond

GPU Speed Of Light Throughput

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 of individual sub-metric of Compute and Memory to clearly identify the highest contributor.

	Throughput [%]	Duration [msecond]
(SM) Throughput [%]	68.45	Duration [msecond]
Throughput [%]	6.38	Elapsed Cycles [cycle]
Throughput [%]	8.38	SM Active Cycles [cycle]
L2 Cache Throughput [%]	0.57	SM Frequency [cycle/nsecond]
DRAM Throughput [%]	0.29	DRAM Frequency [cycle/nsecond]

High Compute Throughput Compute is more heavily utilized than Memory: Look at the [Compute Workload Analysis](#) section to see what the computation is redundant and could be reduced or moved to look-up tables.

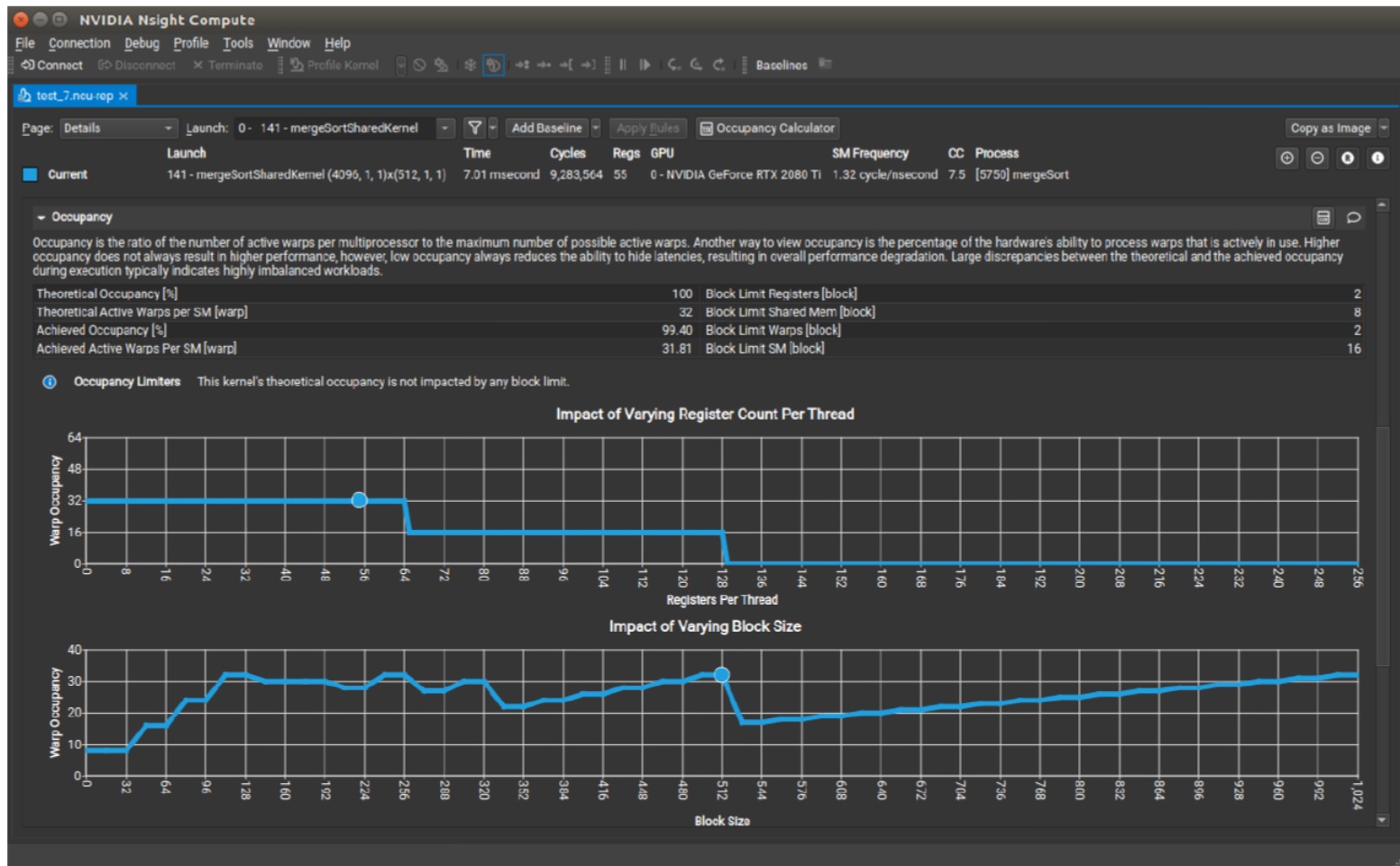
Memory Workload Analysis

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 bandwidth between these units (Max Bandwidth), as breaching the maximum throughput of issuing memory instructions (Max Discrete Queue). Detailed about...

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





Additional Links

- Nsight Systems User Guide
 - <https://docs.nvidia.com/nsight-systems/UserGuide/index.html>
- Nsight Compute
 - User Guide: <https://docs.nvidia.com/nsight-compute/NsightCompute/index.html>
 - Profiling Guide: <https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html>
- Nsight Tutorials
 - <https://developer.nvidia.com/tools-tutorials>
 - [NVIDIA/nsight-training: Training material for Nsight developer tools \(github.com\)](#)
- CUDA GDB User Guide
 - <https://docs.nvidia.com/cuda/cuda-gdb/index.html>
- Compute Sanitizer Documentation
 - <https://docs.nvidia.com/compute-sanitizer/ComputeSanitizer/index.html>
- Example we walked through
 - [openhackathons-org/HPC_Profiler: Profiling with NVIDIA Nsight Tools Bootcamp \(github.com\)](#)

Questions - Contact Me!

- **Email**
 - jdursi@nvidia.com
- **Alliance Federation Staff Office Hours**
 - Will try this for a few months
 - First Tuesday of the month, 3:30 ET (4:30 AT, 2:30 CT, 1:30 MT, 12:30 PT)
 - <https://meet.google.com/fcd-yofg-bbk>
 - 5 Nov, 3 Dec, 7 Jan
 - Bring any questions you have, I'll answer what I can and get back to you on what I can't.

What should we cover next?

- **Advanced Nsight**
 - Multinode, Reports analysis
 - Jupyter-Notebook, VS Code...
- **Profiling tools for Pytorch**
 - Torch profiler, using nsys with torch, common optimizations
- **Other Dev Tools**
 - Compute-sanitizer, cuda-gdb
- **MIG & MPS**
- **Scientific python options**
 - CuPy, Numba, RAPIDS, cuNumeric, JAX
- **AI for PDEs – Modulus**
- **CUDA-Q and cuQuantum**
- **Federated Learning...**



Academic Workshop Series

Faculty Development



Free Training, Leading to Certification as DLI Instructors

- **First Two Weeks of November**
- **Trainings Listed here:**
 - <https://events.nvidia.com/faculty-development-virtual-workshops-higher-ed>
- **Aimed at faculty, but email me if interested**



Questions to Guide Profile Analysis

- What is hot?
 - Can I make it faster, shrink the problem, parallelize it? (Not always...)
 - Reduce precision?
- What is cold?
 - Fill the gaps in the timeline
 - Can I take advantage of unused hardware?
 - Unnecessary dependencies or syncs?

General Optimization Tips

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

General Optimization Tips

- Fuse tiny kernels, copies, memsets.
 - Check out CUDA Graphs
- Overlap/oversubscribe with MPS
- Multi-buffering
 - Don't make everyone wait on the same piece of memory
 - Double, triple buffer
- Avoid moving data back to the CPU
 - Pre-allocate and recycle!
- Minimize managed memory page faults
 - Prefetch!

- 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

