



# NVIDIA DEVELOPER TOOLS

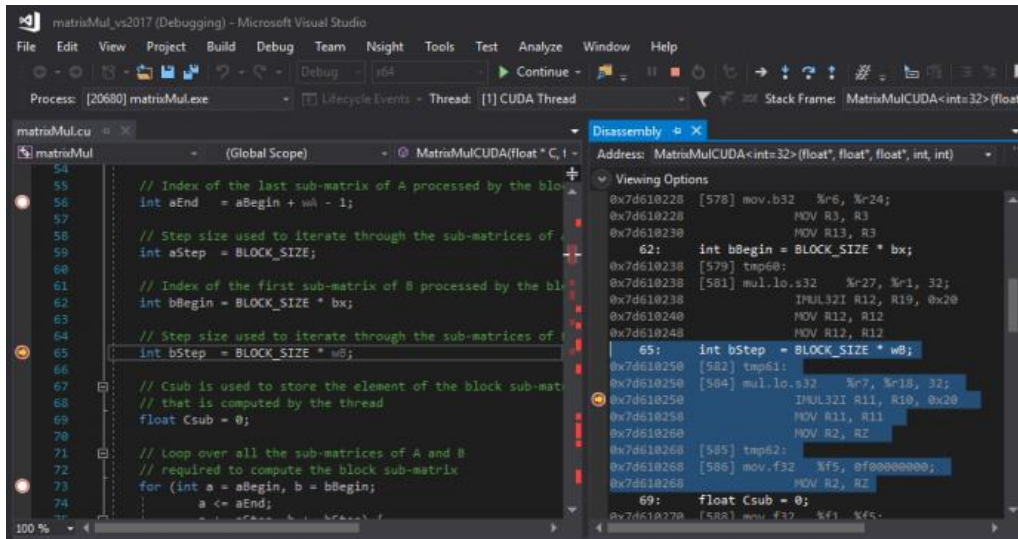
Robert Searles

Senior Solutions Architect, NVIDIA



# Developer Tools Ecosystem

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



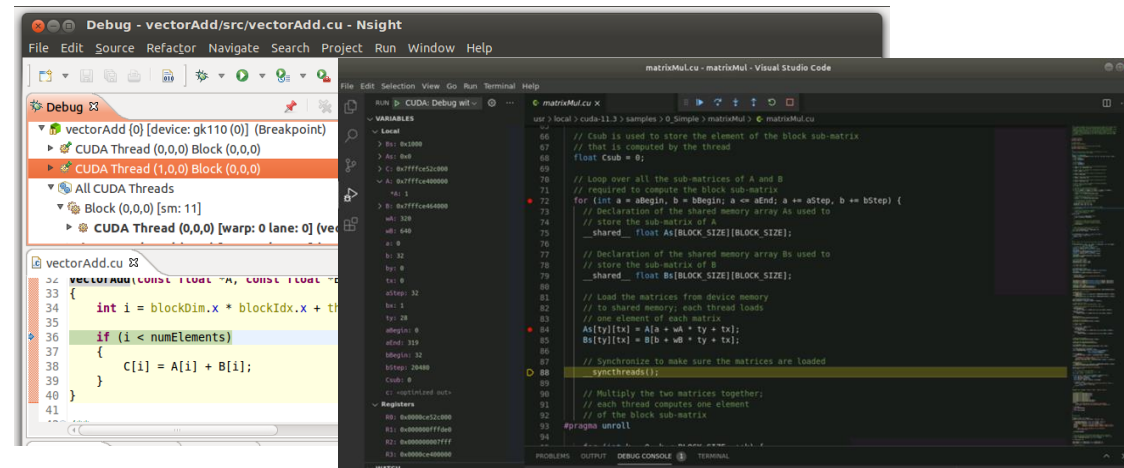
**Profilers:** Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)



**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, Nsight Eclipse Edition



# 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

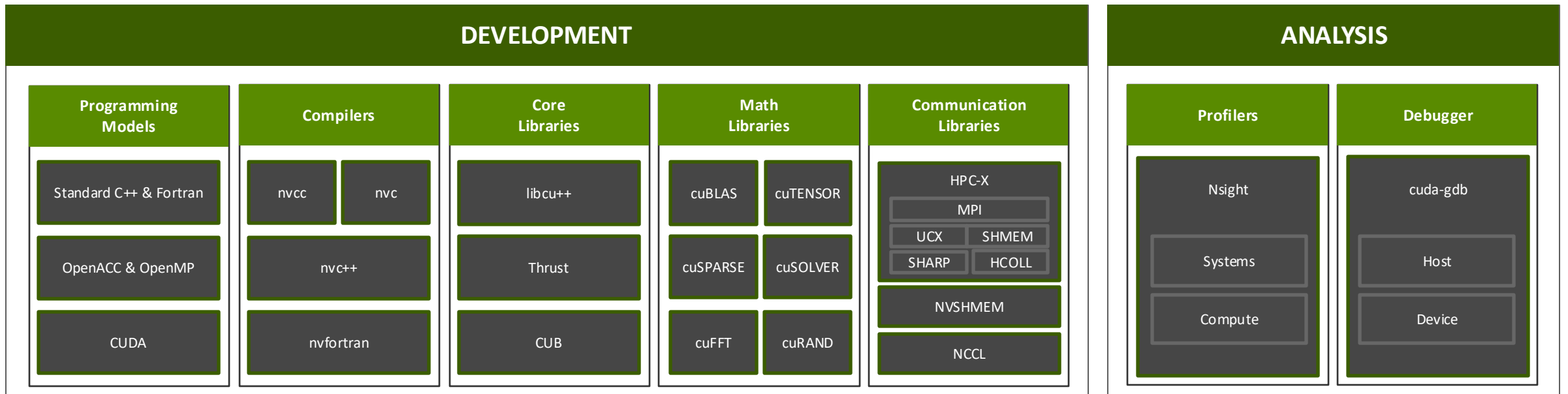
Data Analytics

AI

Quantum

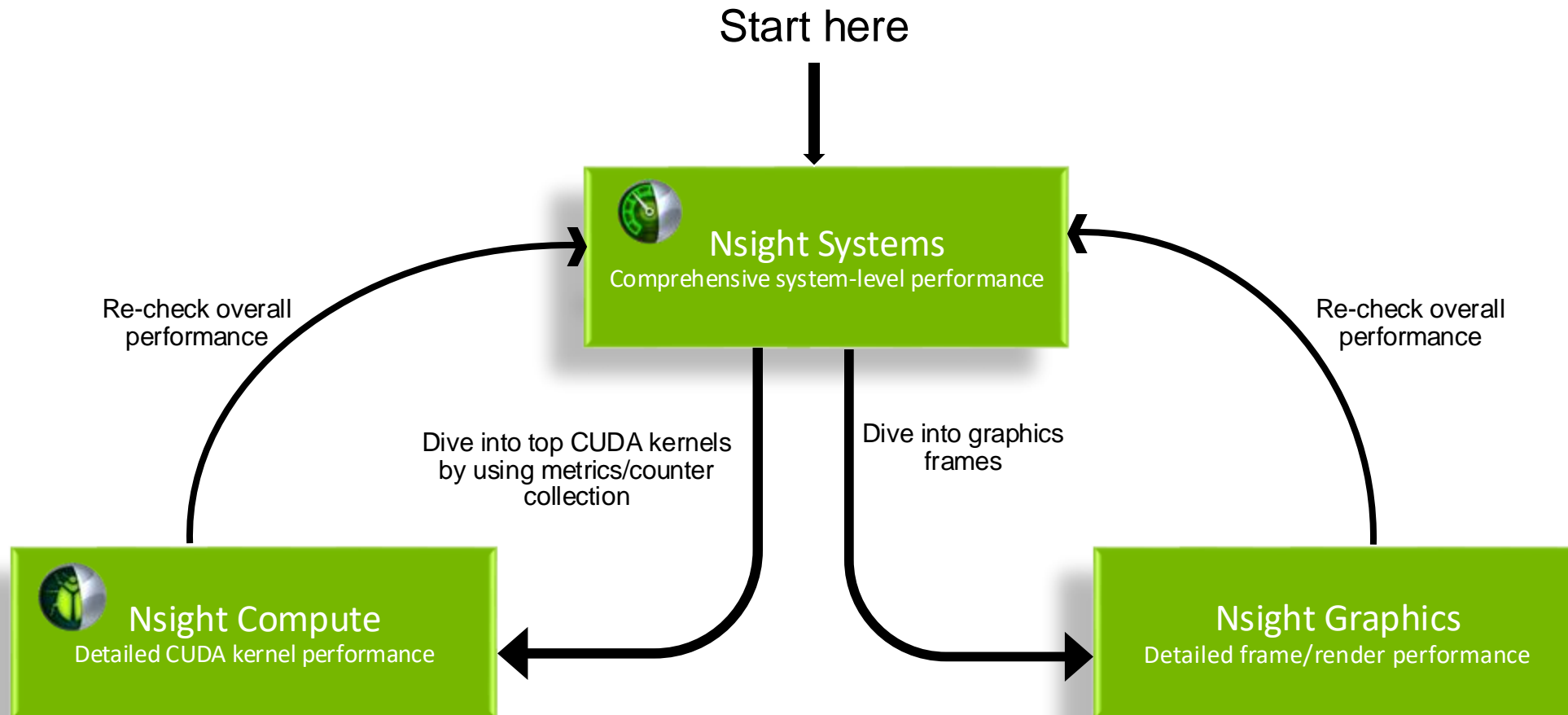
# NVIDIA HPC SDK

Available at [developer.nvidia.com/hpc-sdk](https://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

# NSIGHT Profilers Workflow



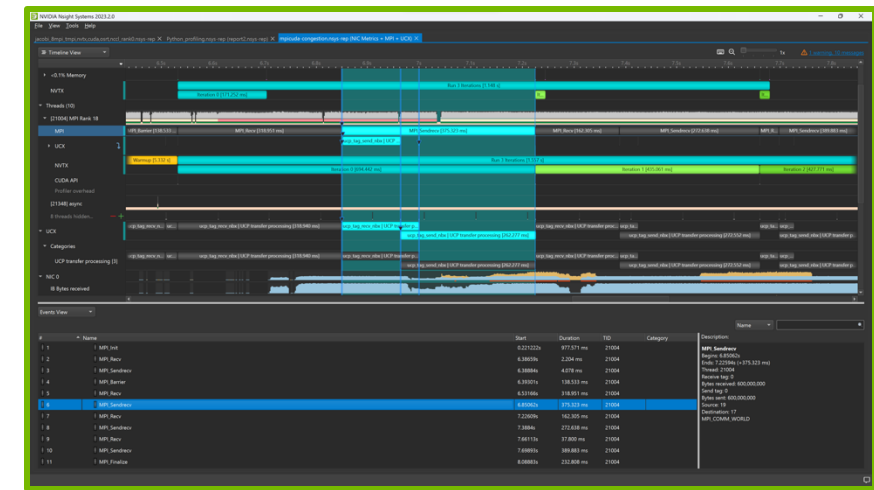
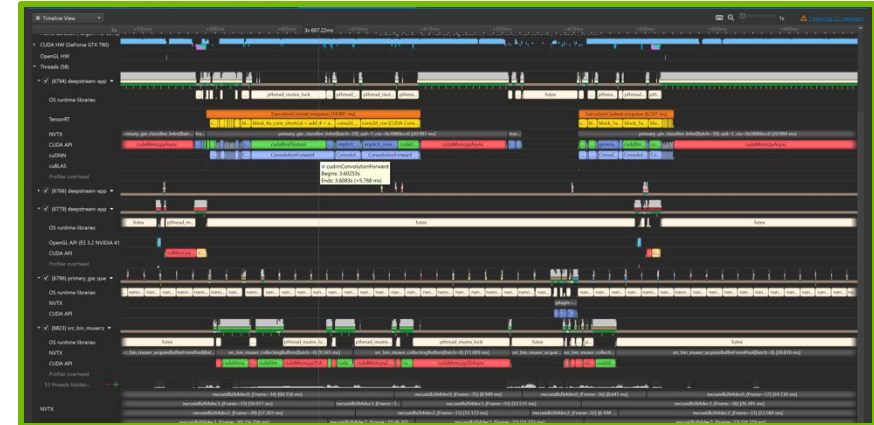


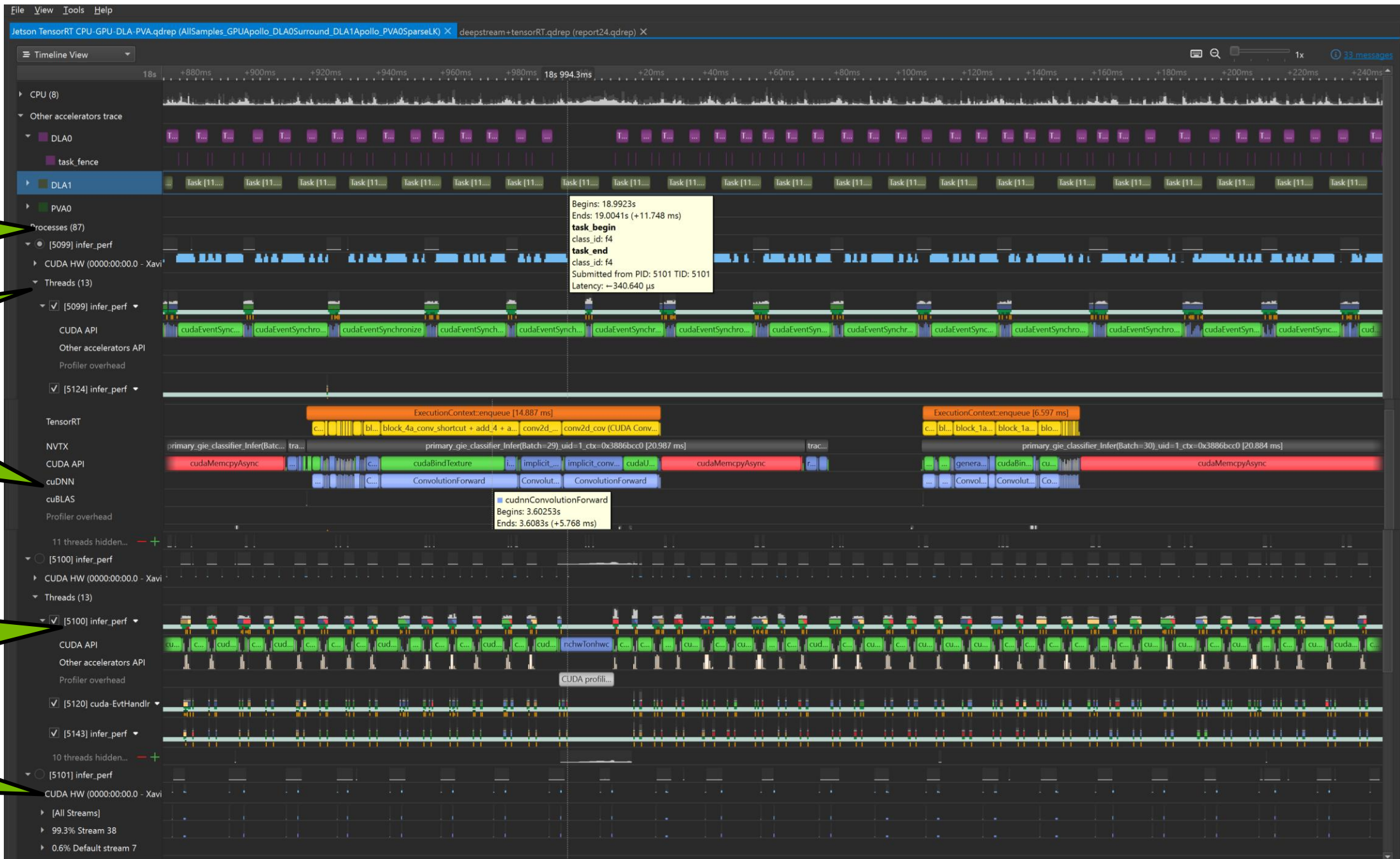
# Nsight Systems

## System Profiler

### Key Features:

- System-wide application algorithm tuning
  - Multi-process tree support
- Locate optimization opportunities
  - Visualize millions of events on a very fast GUI timeline
  - 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 Server, Tegra), Windows, macOS X (host)
- GPUs: Pascal+
- Docs/product: <https://developer.nvidia.com/nsight-systems>





Processes and threads

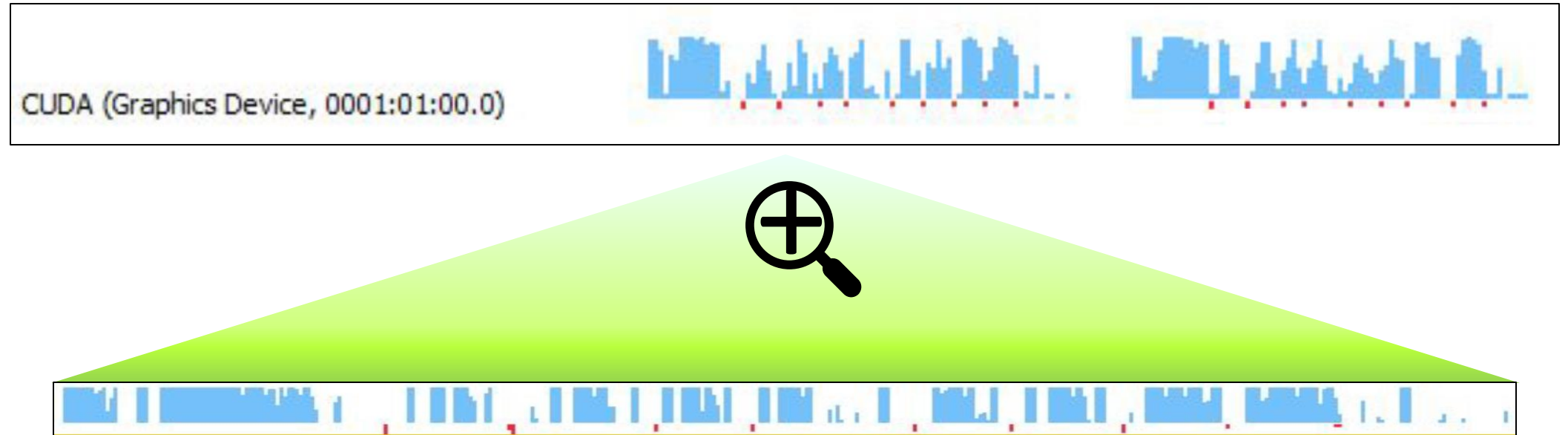
Thread state

cuDNN and cuBLAS trace

Kernel and memory transfer activities

Multi-GPU

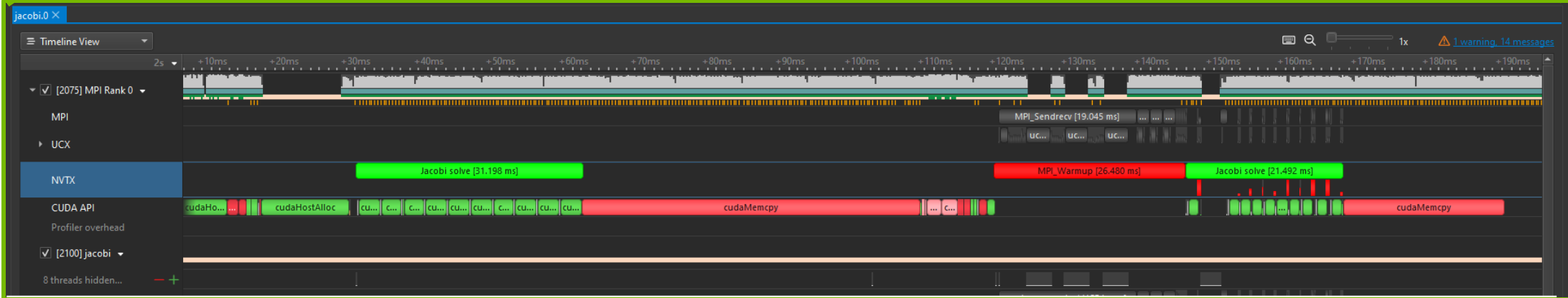
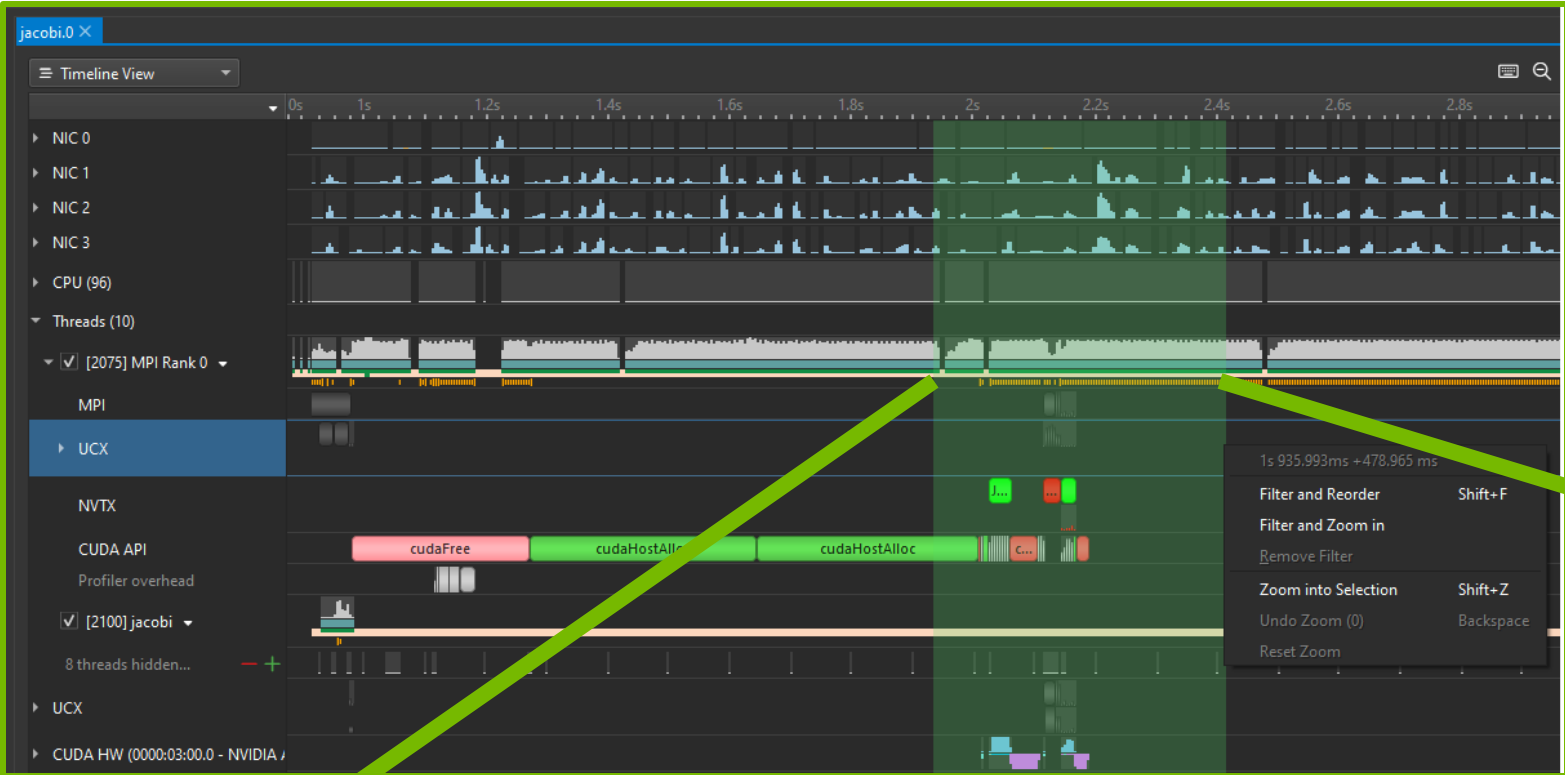
# Reading Utilization



- Zoom in to valleys to find gaps!



# Zoom/Filter to Exact Areas of Interest

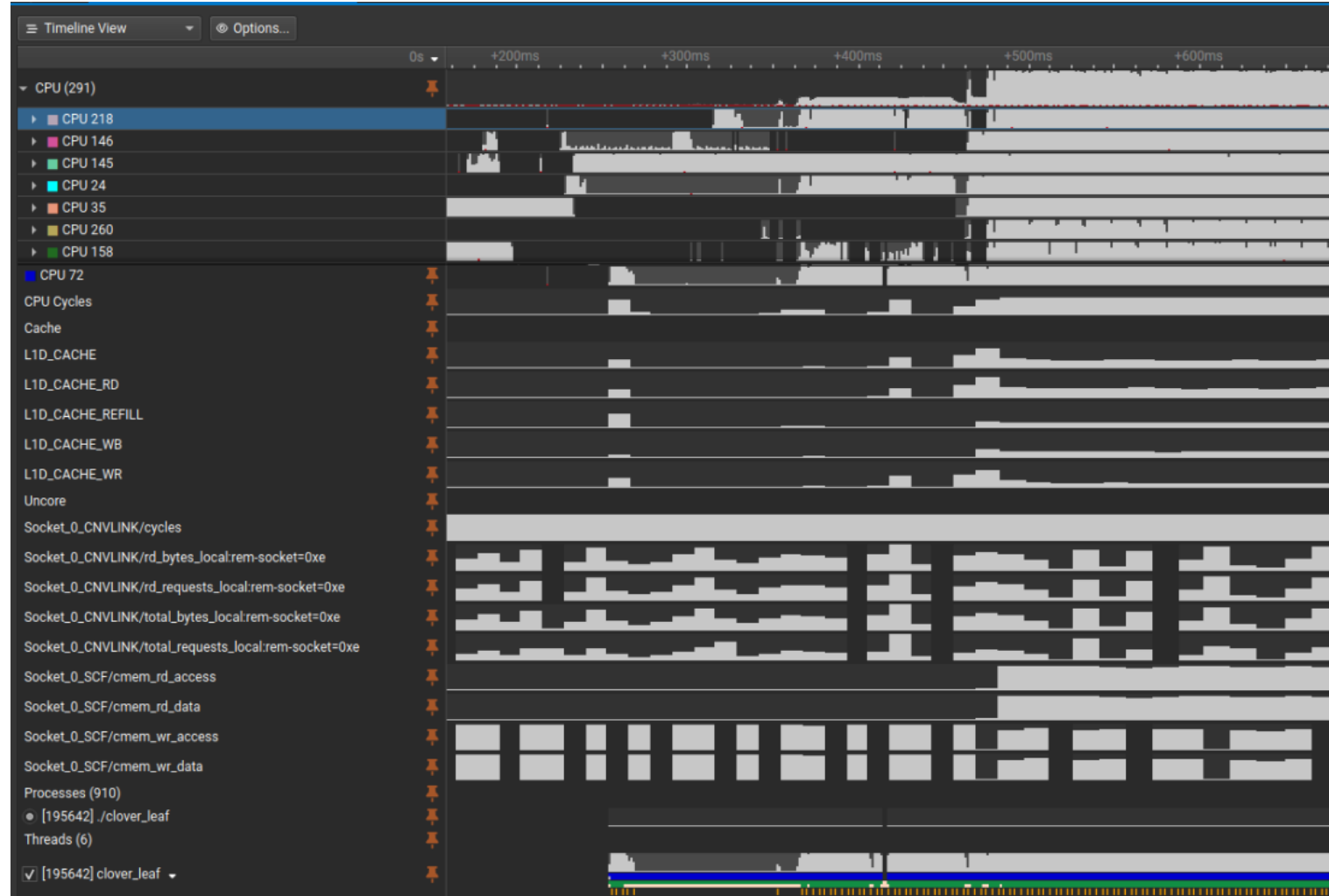




# Grace Host Profiling

## Hardware Counters and Metrics

- CPU Core and Uncore Events
  - Sampled for each CPU
  - Visualize parallelism effects
  - Cache hit/miss, instructions retired, etc...
  - L3 Coherency Fabric
  - Socket to socket traffic
- Variable sampling frequencies supported
- Timeline correlated with all other data
  - GPU vs. CPU idle times and metrics
  - Data movement
  - Zoom and filter

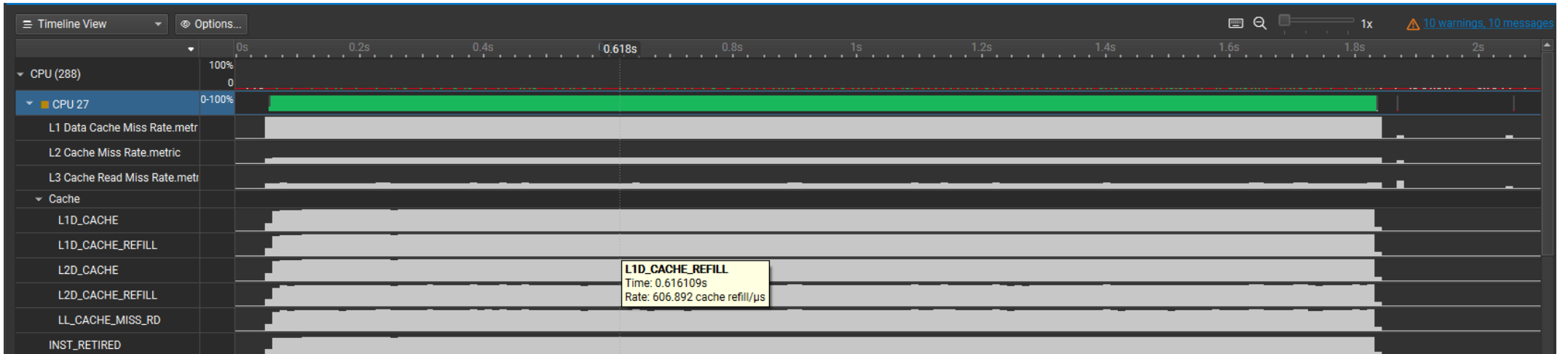




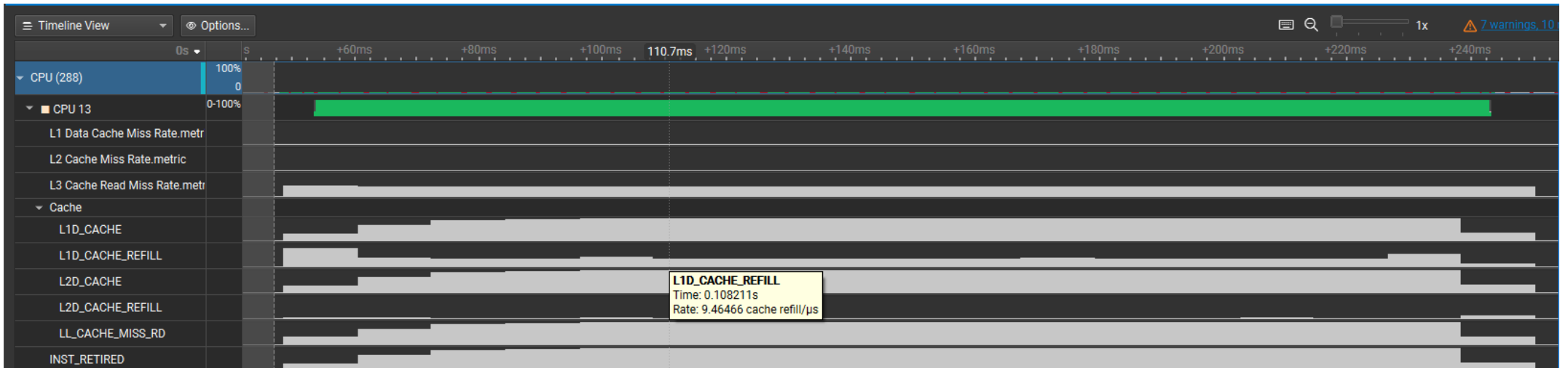
# Grace Host Profiling

## Cache Access Pattern Example

Single threaded CPU matrix multiplication with poor memory access patterns



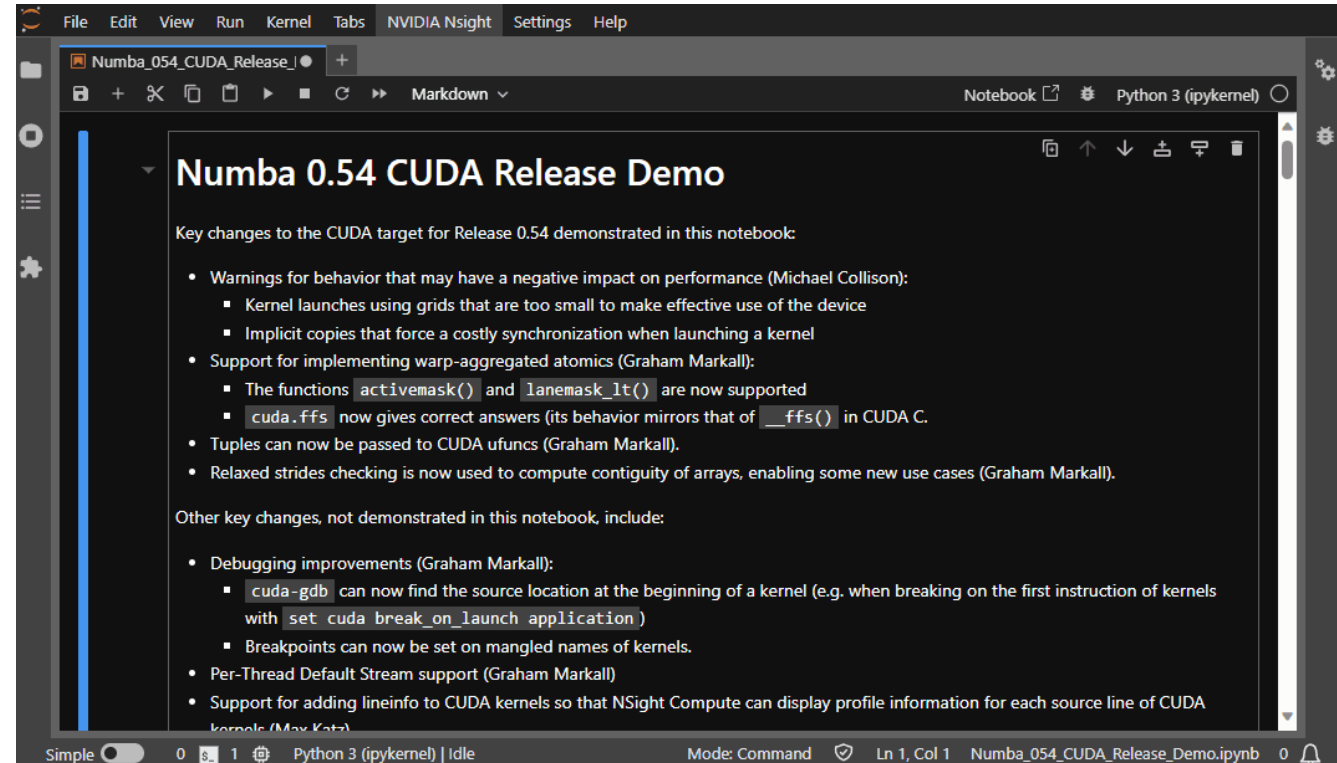
Improving access pattern and implementing cache blocking





# JupyterLab Integration Updates

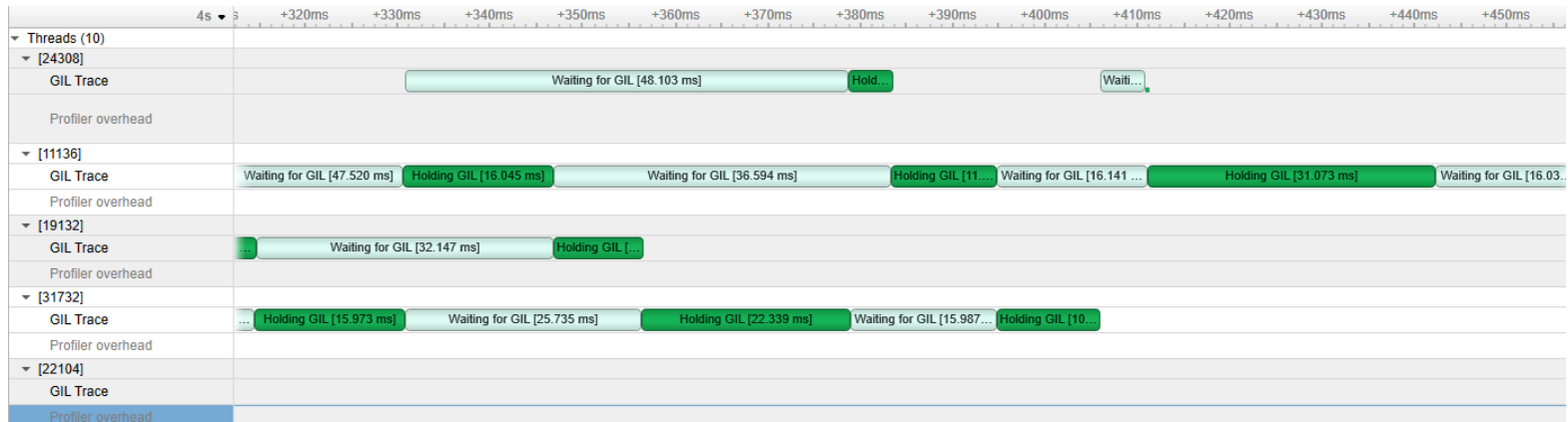
- Extension to JupyterLab
- Profile individual Jupyter cells
- Text-based results can be viewed directly in Jupyter
- Launch **new** remote GUI streaming container directly in JupyterLab
  - Servers without X, Windowing Manager, ...
  - Container with X, WM, & WebRTC server
  - Dockerfile inside Nsight 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...



Timeline range for a CUDA API call

C/C++ frames

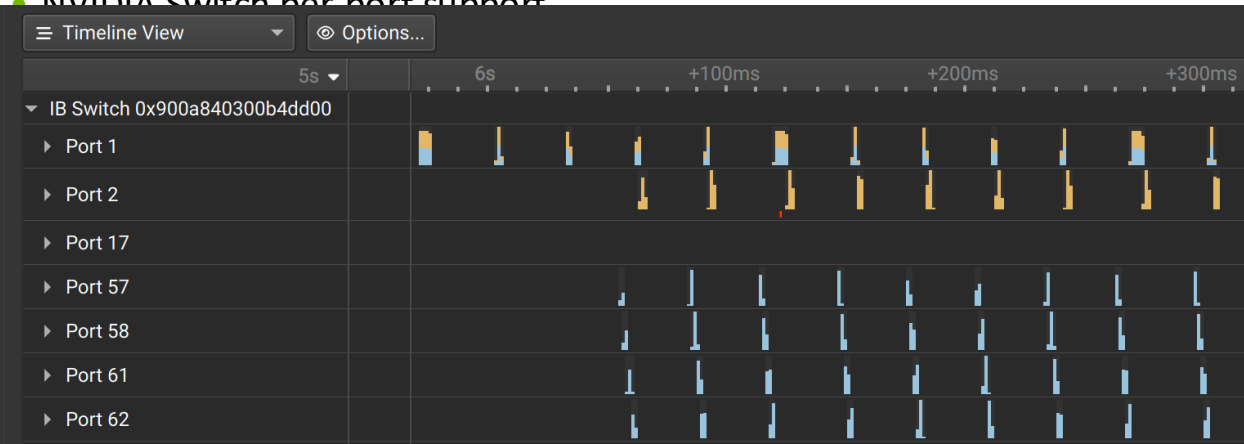
Python frames



# 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 per 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-lqgy7               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-n7rj1                   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    connectivity-agent-7f8fc89f85-c96jx                    2/2     Running   0           3d15h
kube-system    connectivity-agent-7f8fc89f85-stgls                    2/2     Running   0           3d16h
kube-system    connectivity-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-755c7dfdf5-9b5bv                   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-e8d740d6-2jhb 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
kube-system    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$

```

# 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:

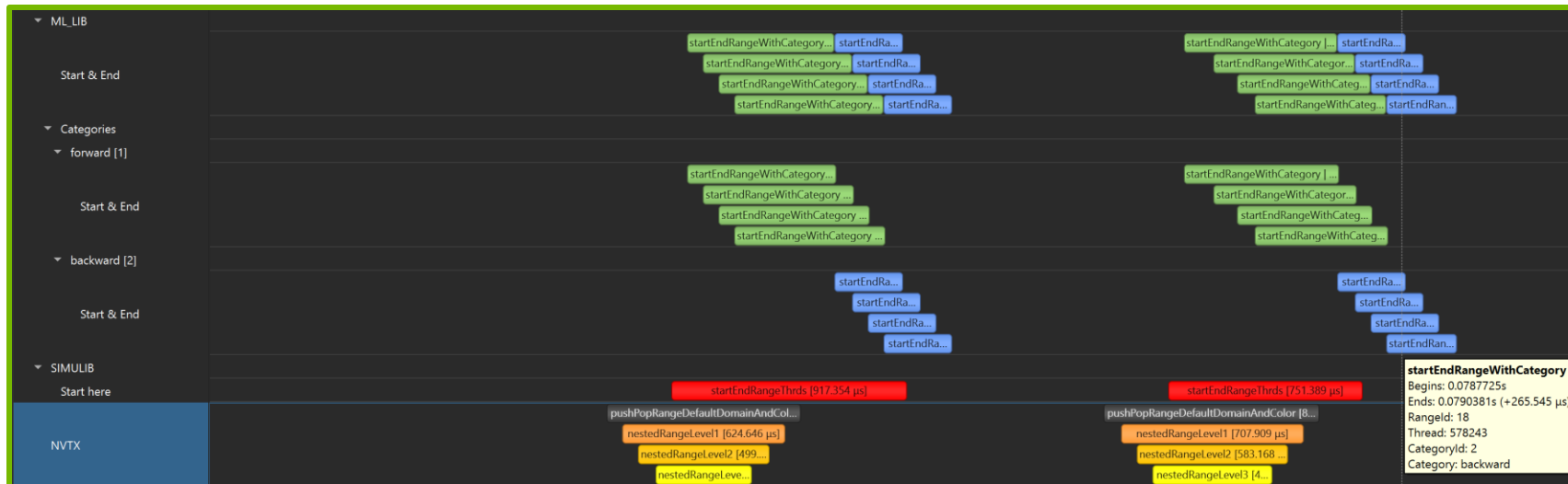
```
nvtxMark("This is a marker");
```

- 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);
```



# Python and NVTX

- Annotate Python code with NVTX

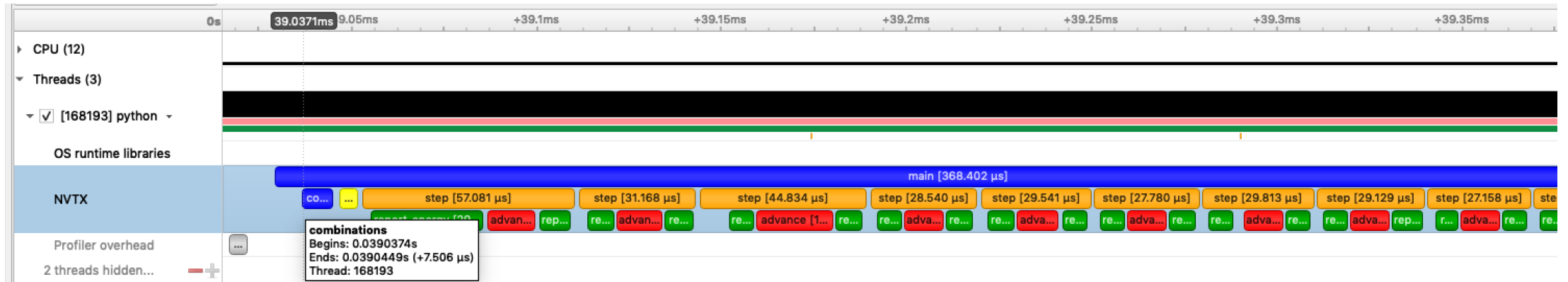
```
# demo.py

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)

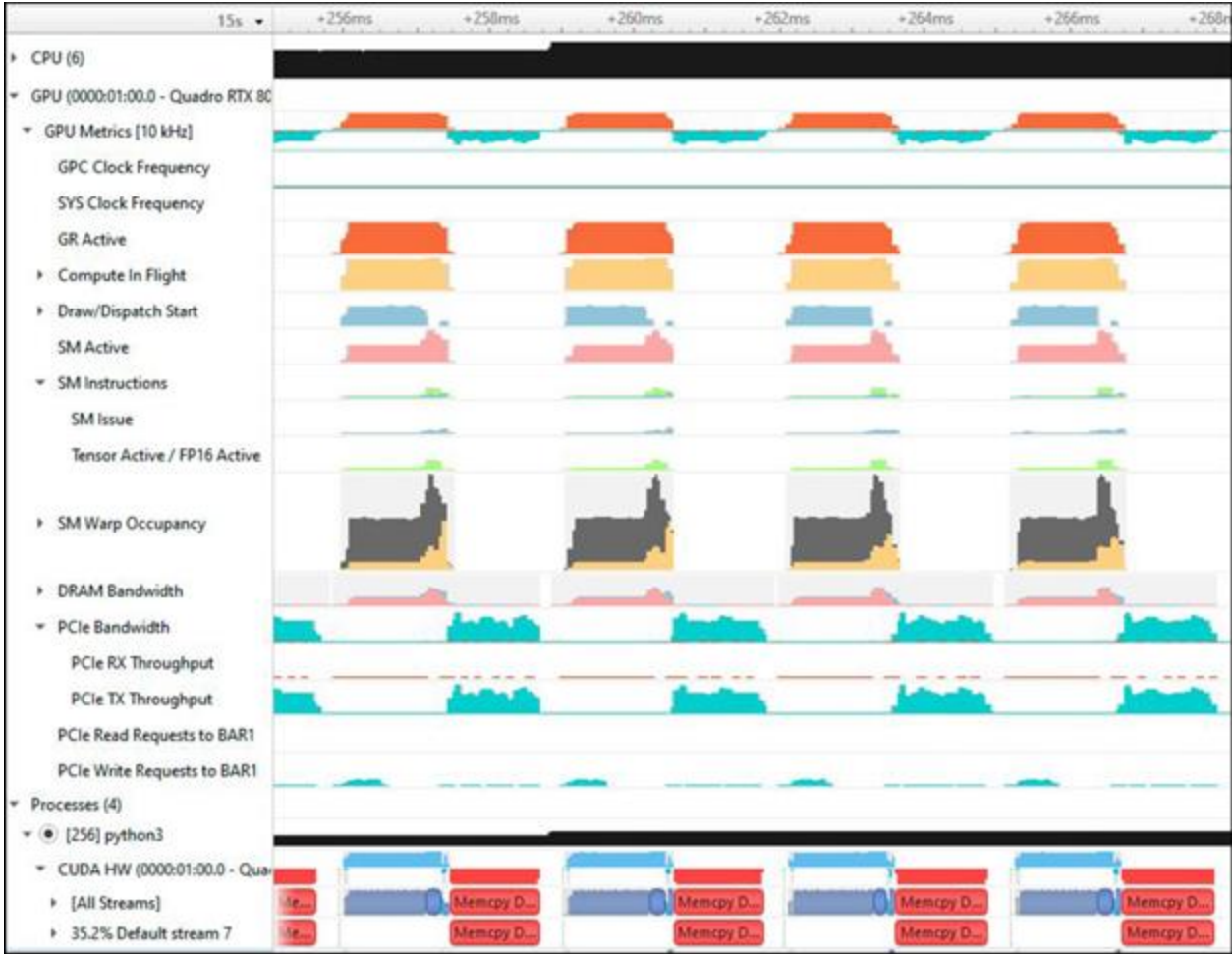
my_function()
```

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





# 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

# Application Profiles with Nsight Systems

```
$ nsys profile -o report -stats=true ./myapp.exe
```

- Generated file: report.qdrep (or report.nsys-rep)  
Open for viewing in the Nsight Systems UI
- When using MPI, recommended to use *nsys* after *mpirun*/*srun*:  

```
$ mpirun -n 4 nsys profile ./myapp.exe
```

# Profiling DL Models

- Pytorch
  - DNN Layer annotations are disabled by default
  - ++ "with `torch.autograd.profiler.emit_nvtx():`"
  - Manually with `torch.cuda.nvtx.range_(push/pop)`
  - TensorRT backend is already annotated
  
- Tensorflow
  - Annotated by default with NVTX in NVIDIA TF containers
  - `TF_DISABLE_NVTX_RANGES=1` to disable for production

# 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!

# Expert Systems & Statistics

Built-in data analytics with advice

The screenshot displays the NVIDIA Nsight Systems interface. The top section shows a timeline view of system events. A callout menu is open, listing various system events such as 'CUDA Async Memcpy with Pageable Memory', 'CUDA Synchronous Memcpy', and 'CUDA Synchronization APIs'. Below the timeline, the 'Expert System View' is active, showing a table of events and associated advice.

Event Name	Duration	Start	Src Kind	Dst Kind	Bytes	PID	Device ID	Context ID	Stream ID	API Name
CUDA Async Memcpy with Pageable Memory	2,048 µs	6,38792s	Device	Pageable	8 B	75475		0	1	7 cudaMemcpy
The following APIs use PAGEABLE memory which causes asynchronous CUDA memcpy operations to block and be executed synchronously. This leads to low GPU utilization.	2,048 µs	6,8334s	Device	Pageable	4 B	75475		0	1	7 cudaMemcpy
	2,016 µs	2,5394s	Device	Pageable	4 B	75475		0	1	7 cudaMemcpy
	2,016 µs	3,90617s	Device	Pageable	48 B	75475		0	1	7 cudaMemcpy
Suggestion: If applicable, use PINNED memory instead.	2,016 µs	4,25257s	Device	Pageable	4 B	75475		0	1	7 cudaMemcpy
	2,016 µs	5,67617s	Device	Pageable	48 B	75475		0	1	7 cudaMemcpy
	2,016 µs	5,9572s	Device	Pageable	8 B	75475		0	1	7 cudaMemcpy
	2,016 µs	5,97088s	Device	Pageable	4 B	75475		0	1	7 cudaMemcpy

CLI command:  
nsys analyze -r cuda-async-memcpy /mnt/data/traces/qdrep/ncc1/profile\_circe-n011\_506451\_0.sqlite

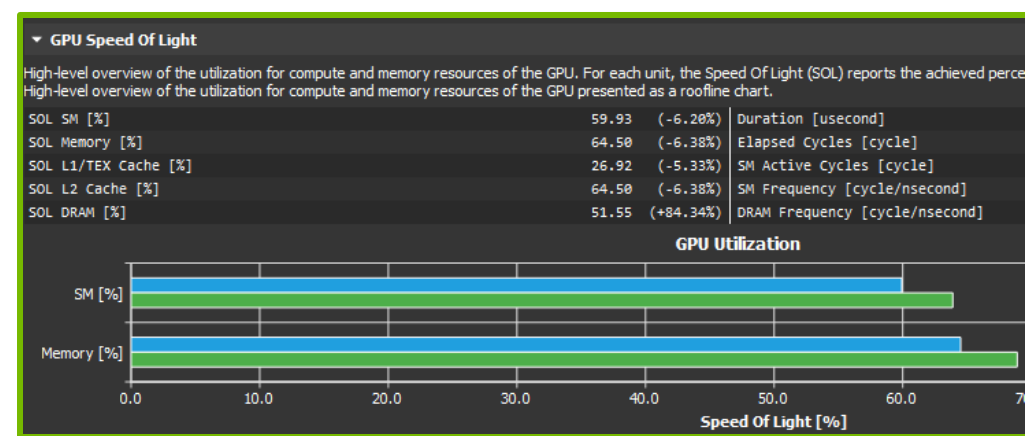


# Nsight Compute

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>



Metric	Value
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



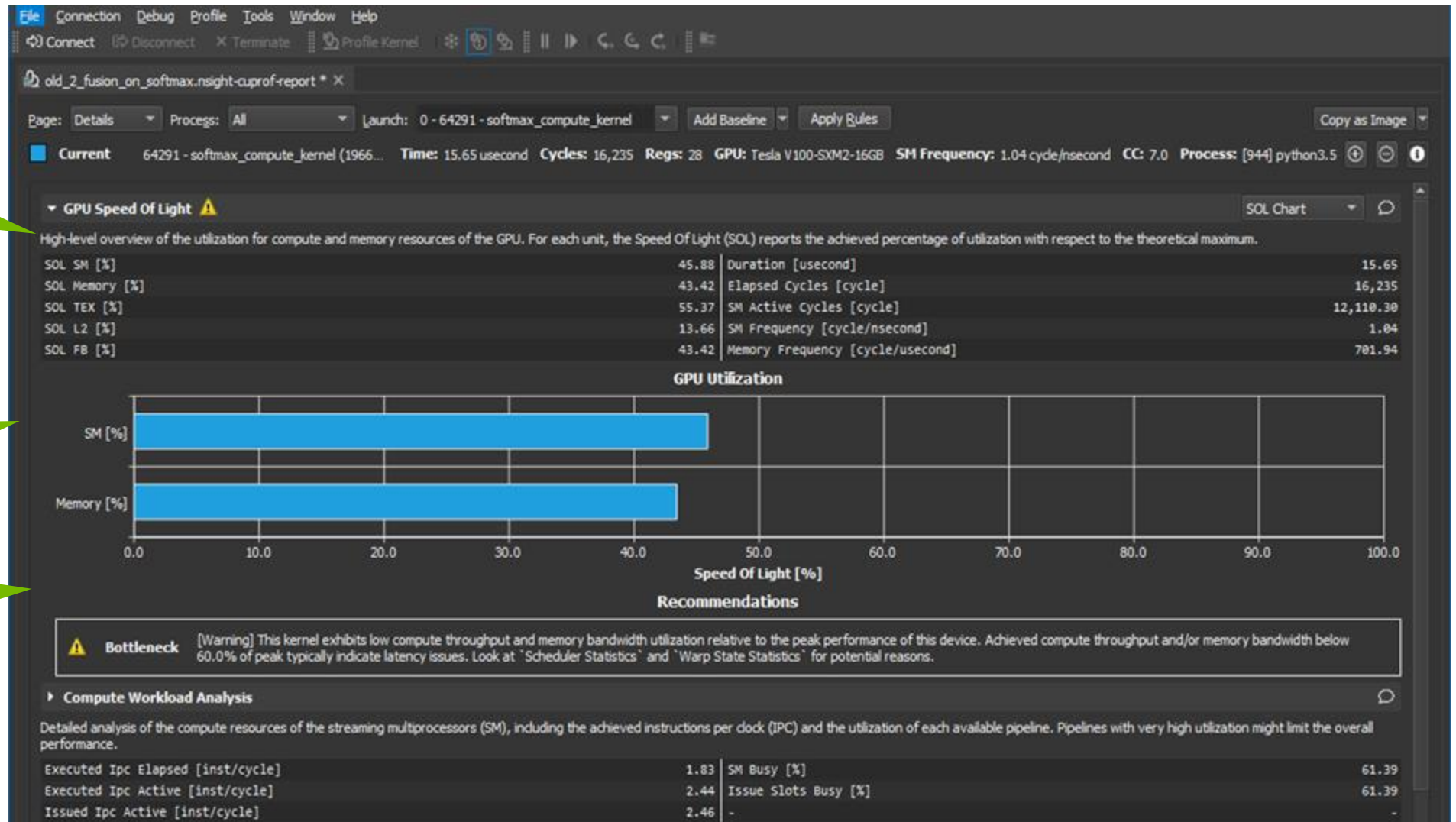


# Nsight Compute GUI Interface

Targeted metric sections

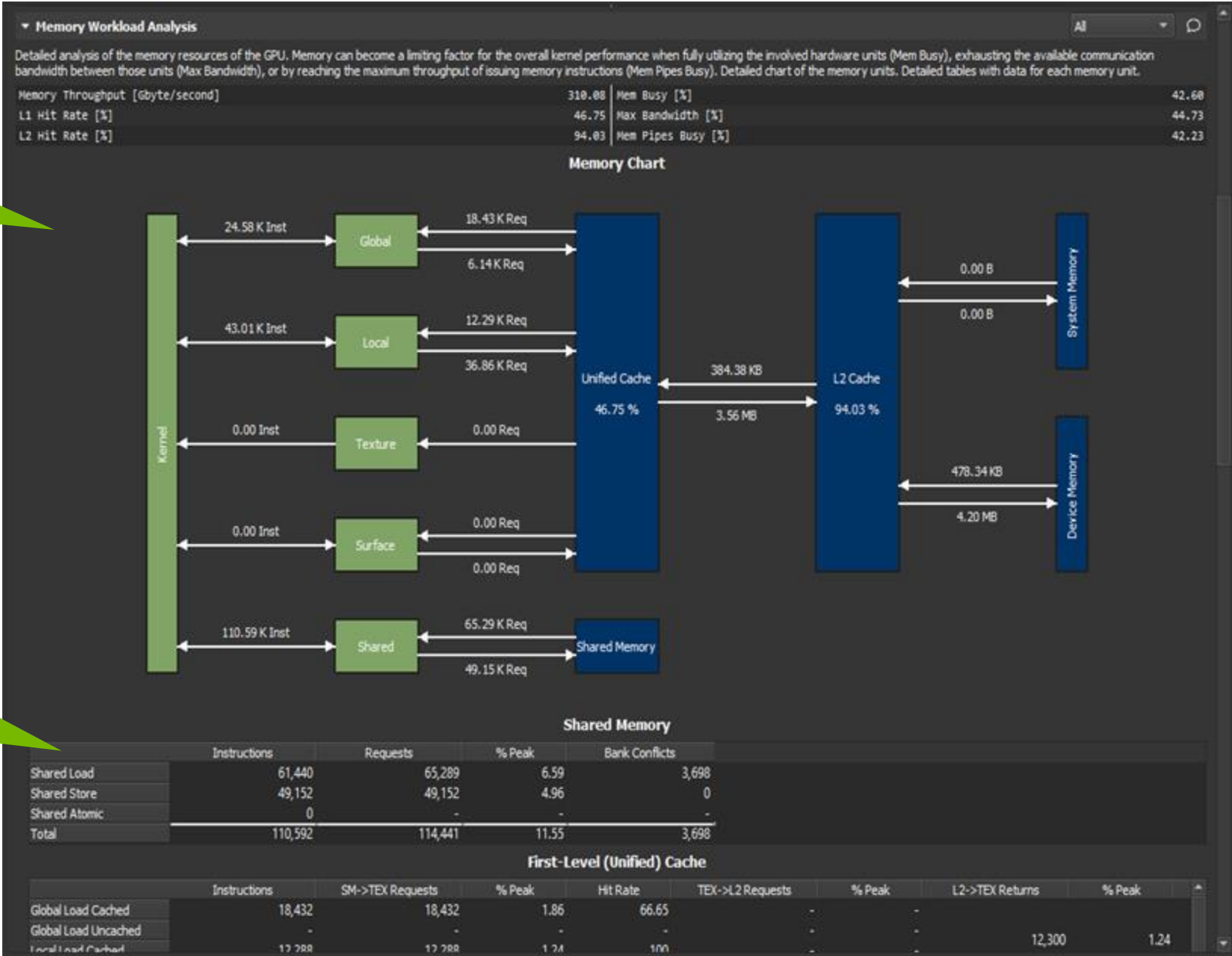
Customizable data collection and presentation

Built-in expertise for Guided Analysis and optimization



Visual memory analysis chart

Metrics for peak performance ratios



Instructions Executed

#	Source	Sampling Data (All)
234	typename DType, typename OType>	0
235	__global__ void softmax_compute_kernel(DType *in, OType *out, index_t M	0
236	Shape<ndim> sshape, Shape<ndim>	0
237	const double temperature) {	0
238	const unsigned x_size = 1 << x_bits;	0
239	__shared__ AType smem[x_size];	0
240	index_t sa = stride[axis];	61
241	index_t base = unravel_dot(blockIdx.x, sshape, stride);	0
242	index_t x = threadIdx.x;	52
243		0
244	red::maximum::SetInitValue(smem[x]);	44
245	for (index_t i = x; i < M; i += x_size) {	0
246	smem[x] = ::max(smem[x], negate ? -in[base + i*sa] : in[base + i*sa	0
247	}	0
248	__syncthreads();	0
249	cuda::Reduce1D<red::maximum, x_bits>(smem);	111
250		0
251	(@);	0
252	(value(smem[x]));	0
253		0
254	DType val;	14
255	for (index_t i = x; i < M; i += x_size) {	0
256	val = negate ? -in[base + i*sa]:in[base + i*sa];	118
257	val = negate ? -in[base + i*sa]:in[base + i*sa];	0
258	smem[x] += static_cast<AType>(expf((val - smax) / static_cast<AType	0
259	}	0
260	__syncthreads();	0
261	cuda::Reduce1D<red::sum, x_bits>(smem);	208
262	__syncthreads();	0
263	AType ssum = smem[x];	0
264	__syncthreads();	0
265		0
266	for (index_t i =	0
267	val = negate ? -in[base + i*sa] : in[base + i*sa];	14
268	out[base + i*sa] = OP::Map((val - smax)/static_cast<DType>(temperat	0
269	}	0

Sampling Data (All)

#	Source	Sampling Data (All)	Instructions Executed
133	BSYNC B0	0	6,144
134	NOP	0	6,144
135	BAR.SYNC 0x0	1	6,144
136	ISETP.GT.AND P0, PT, R11, 0x3f, PT	2	6,144
137	BSSY B1, 0x7f87d326fc50	1	6,144
138	ISETP.GT.AND P1, PT, R11, 0x1f, PT	1	6,144
139	ISETP.GT.AND P2, PT, R11, 0xf, PT	0	6,144
140	ISETP.GT.AND P3, PT, R11, 0x7, PT	2	6,144
141	@!P0 LDS.U R4, [R14+0x100]	2	6,144
142	@!P0 LDS.U R5, [R14]	4	6,144
143	@!P0 STL [R1+0x8], R4	3	6,144
144	@!P0 FMMX R5, R5, R4, !PT	2	6,144
145	@!P0 STS [R14], R5	4	6,144
146	NOP	0	6,144
147	BAR.SYNC 0x0	4	6,144
148	@!P1 LDS.U R6, [R14+0x80]	8	6,144
149	ISETP.GT.AND P0, PT, R11, 0x3, PT	0	6,144
150	P1 LDS.U R7, [R14]	1	6,144
151	P1 STL [R1+0xc], R6	4	6,144
152	P1 FMMX R7, R7, R6, !PT	0	6,144
153	P1 STS [R14], R7	3	6,144
154	NOP	4	6,144
155	BAR.SYNC 0x0	0	6,144
156	@!P2 LDS.U R8, [R14+0x40]	4	6,144
157	ISETP.GT.AND P1, PT, R11, 0x1, PT	0	6,144
158	@!P2 LDS.U R9, [R14]	2	6,144
159	@!P2 STL [R1+0x10], R8	0	6,144
160	@!P2 FMMX R9, R9, R8, !PT	0	6,144
161	@!P2 STS [R14], R9	0	6,144
162	NOP	1	6,144
163	@!P3 LDS.U R10, [R14+0x20]	1	6,144
164	@!P3 LDS.U R5, [R14]	0	6,144
165	@!P3 STL [R1+0x14], R10	4	6,144
166	@!P3 FMMX R5, R5, R10, !PT	2	6,144
167	@!P3 STS [R14], R5	2	6,144
168	NOP	0	6,144

Sampling Data (All)

Instructions Executed

Total Sample Count: 111  
 Barrier: 43 (38.7%)  
 Mio Throttle: 21 (18.9%)  
 Not Selected: 8 (7.2%)  
 Selected: 7 (6.3%)  
 Short Scoreboard: 16 (14.4%)  
 Wait: 16 (14.4%)

Source/PTX/SASS analysis and correlation

Source metrics per instruction

Metric heatmap to quickly identify hotspots

## 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)

# Standalone Source Viewer

- View of side-by-side assembly and correlated source code for CUDA kernels
- No profile required
- Open .cubin files directly
- Helps identify compiler optimizations and inefficiencies

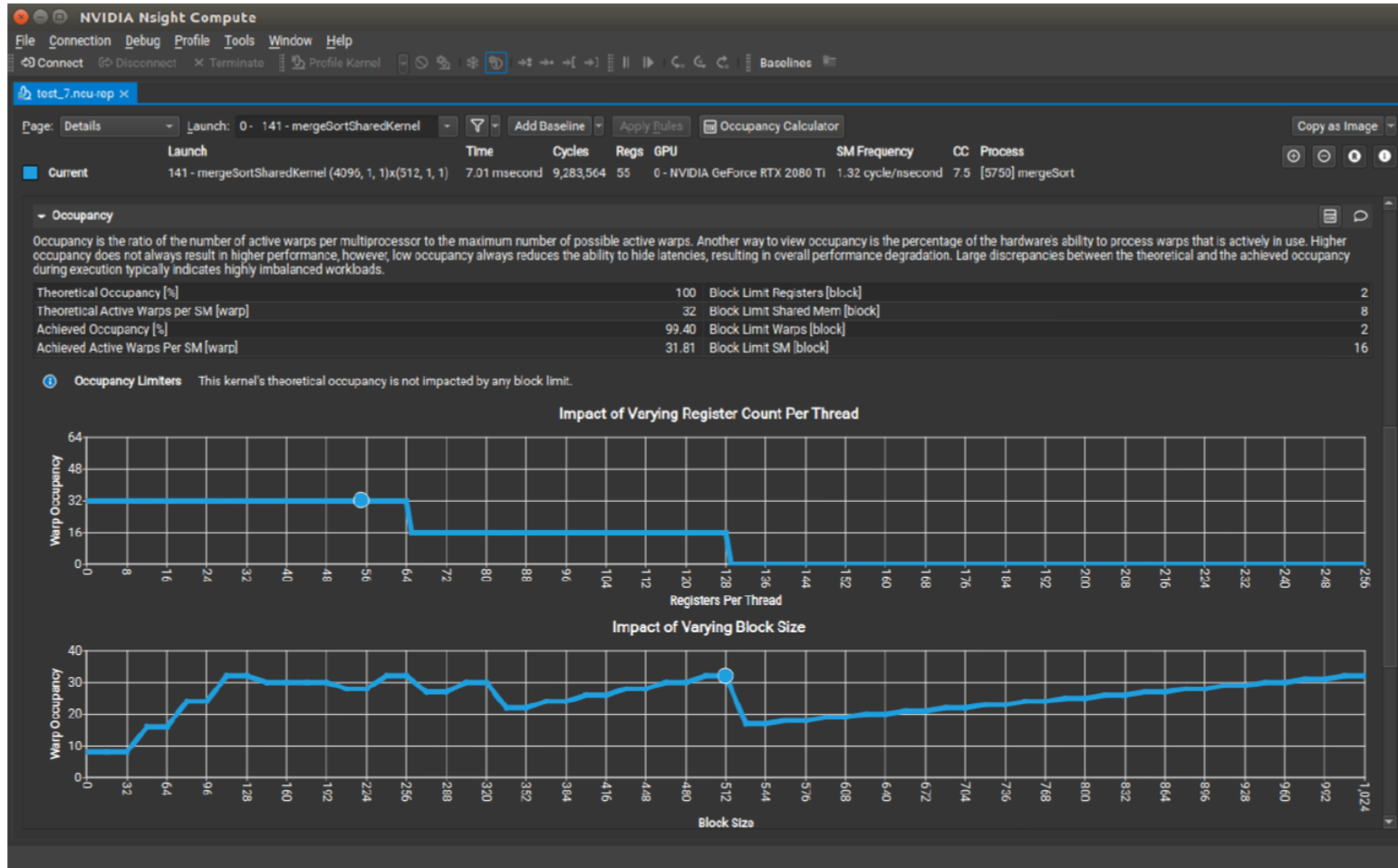
The screenshot displays the Standalone Source Viewer interface. The top bar shows the current launch configuration: "Launch: 0 -32655 - device\_tea\_leaf\_ppcg\_sol" and "Current" launch details for "32655 - device\_tea\_leaf\_ppcg\_solve\_update\_r (126, 1001, 1)x(32, 4, 1)" with a time of 1.07 msecond, 1,458,003 cycles, 32 registers, and GPU NVIDIA GeForce RTX 2080 TI. The interface is split into two main panels. The left panel shows the source code for the kernel "tea\_leaf\_ppcg\_cuknl", with line 165 highlighted in green. The right panel shows the corresponding assembly code for "device\_tea\_leaf\_ppcg\_solve\_update\_r", with instructions 22 through 46 visible. The assembly instructions include operations like IADD3, MOV, IMAD, IMAD.WIDE, LDG.E.64.CONSTANT.SYS, and DADD.

```
Page: Source Launch: 0 -32655 - device_tea_leaf_ppcg_sol Add Baseline Apply Rules
Current 32655 - device_tea_leaf_ppcg_solve_update_r (126, 1001, 1)x(32, 4, 1) 1.07 msecond 1,458,003 32 NVIDIA GeForce RTX 2080 TI 1.36 cycle/msecond 7.5 [10906] tea_leaf
View: Source and SASS
Source: tea_leaf_ppcg_cuknl Find... Navigation: Instructions Executed
# Address Source
145 sd[THARR2D(0, 0, 0)] = alpha[step]*sd[THARR2D(0, 0, 0)]
146 + beta[step]*r[THARR2D(0, 0, 0)];
147 }
148 }
149 }
150 }
151 }
152 /* New update to rtemp for use in calc_sd */
153
154 __global__ void device_tea_leaf_ppcg_solve_update_r
155 (kernel_info_t kernel_info,
156 double * __restrict const rtemp,
157 const double * __restrict const Kx,
158 const double * __restrict const Ky,
159 const double * __restrict const sd)
160 {
161     __kernel_indexes;
162
163     if (WITHIN_BOUNDS)
164     {
165         const double result = (1.0
166 + (Ky[THARR2D(0, 1, 0)] + Ky[THARR2D(0, 0, 0)])
167 + (Kx[THARR2D(1, 0, 0)] + Kx[THARR2D(0, 0, 0)])) * sd[THARR2D(0, 0, 0)]
168 - (Ky[THARR2D(0, 1, 0)] * sd[THARR2D(0, 1, 0)] + Ky[THARR2D(0, 0, 0)] * sd[THARR2D(0, -1, 0)])
169 - (Kx[THARR2D(1, 0, 0)] * sd[THARR2D(1, 0, 0)] + Kx[THARR2D(0, 0, 0)] * sd[THARR2D(-1, 0, 0)]);
170
# Address Source
22 00007f72 42fa6a59 IADD3 R5, R3, 0x1, R2
23 00007f72 42fa6a69 MOV R20, 0x8
24 00007f72 42fa6a79 IMAD R21, R3, R2, R0
25 00007f72 42fa6a89 IMAD R5, R2, R5, R0
26 00007f72 42fa6a99 IMAD R3, R3, R2, -R2
27 00007f72 42fa6aa9 IMAD.WIDE R16, R21, R20, c[0x0][0x1a0]
28 00007f72 42fa6ab9 IADD3 R3, R0, R3, R2
29 00007f72 42fa6ac9 IMAD.WIDE R10, R5, R20, c[0x0][0x1a0]
30 00007f72 42fa6ad9 IMAD.WIDE R18, R3, R20, c[0x0][0x1a0]
31 00007f72 42fa6ae9 LDG.E.64.CONSTANT.SYS R16, [R16]
32 00007f72 42fa6af9 IMAD.WIDE R26, R21, R20, c[0x0][0x190]
33 00007f72 42fa6b09 LDG.E.64.CONSTANT.SYS R10, [R10]
34 00007f72 42fa6b19 IMAD.WIDE R28, R21, R20, c[0x0][0x1a0]
35 00007f72 42fa6b29 LDG.E.64.CONSTANT.SYS R18, [R18]
36 00007f72 42fa6b39 IMAD.WIDE R12, R5, R20, c[0x0][0x1a0]
37 00007f72 42fa6b49 LDG.E.64.CONSTANT.SYS R14, [R26]
38 00007f72 42fa6b59 LDG.E.64.CONSTANT.SYS R6, [R26+0x8]
39 00007f72 42fa6b69 LDG.E.64.CONSTANT.SYS R2, [R28+0x0]
40 00007f72 42fa6b79 LDG.E.64.CONSTANT.SYS R12, [R12]
41 00007f72 42fa6b89 LDG.E.64.CONSTANT.SYS R8, [R28+0x8]
42 00007f72 42fa6b99 LDG.E.64.CONSTANT.SYS R4, [R28]
43 00007f72 42fa6ba9 IMAD.WIDE R20, R21, R20, c[0x0][0x190]
44 00007f72 42fa6bb9 LDG.E.64.SYS R22, [R20]
45 00007f72 42fa6bc9 DADD R24, R16, R10
46 00007f72 42fa6bd9 DMIII R18, R16, R18
```

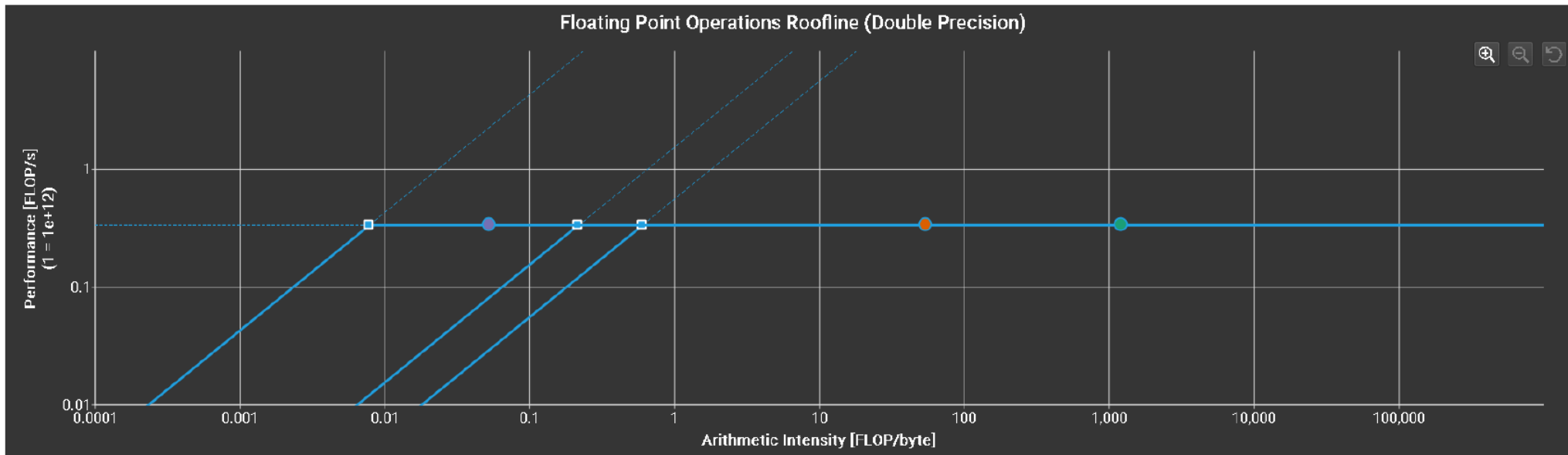
# Occupancy Calculator

Model hardware usage and identify limiters

- Model theoretical hardware usage
- Understand limitations from hardware vs. kernel parameters
- Configure model to vary HW and kernel parameters
- Opened from an existing report or as a new activity



# Hierarchical Roofline



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

Sections/Rules Info

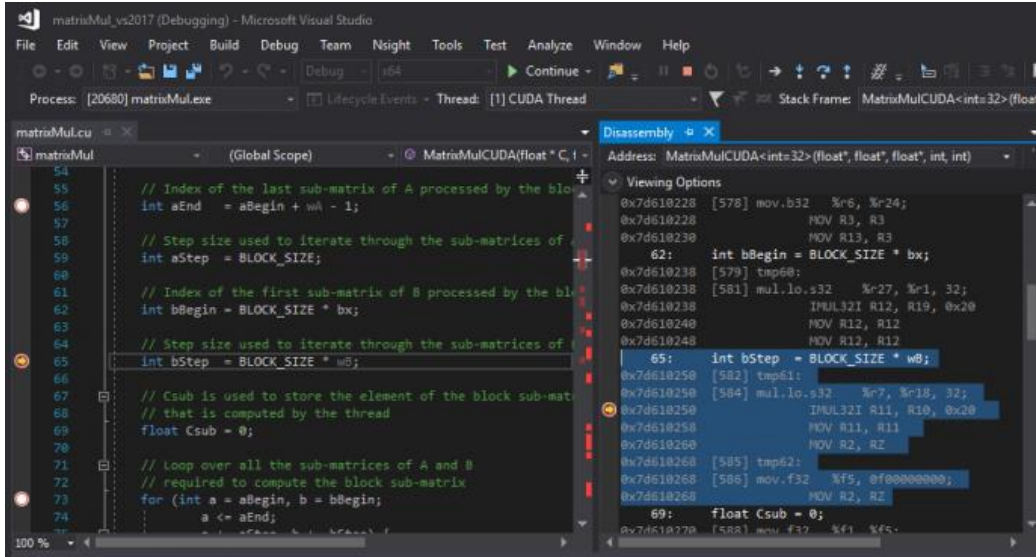
Sections/Rules   Enable All  Disable All

Enter filter

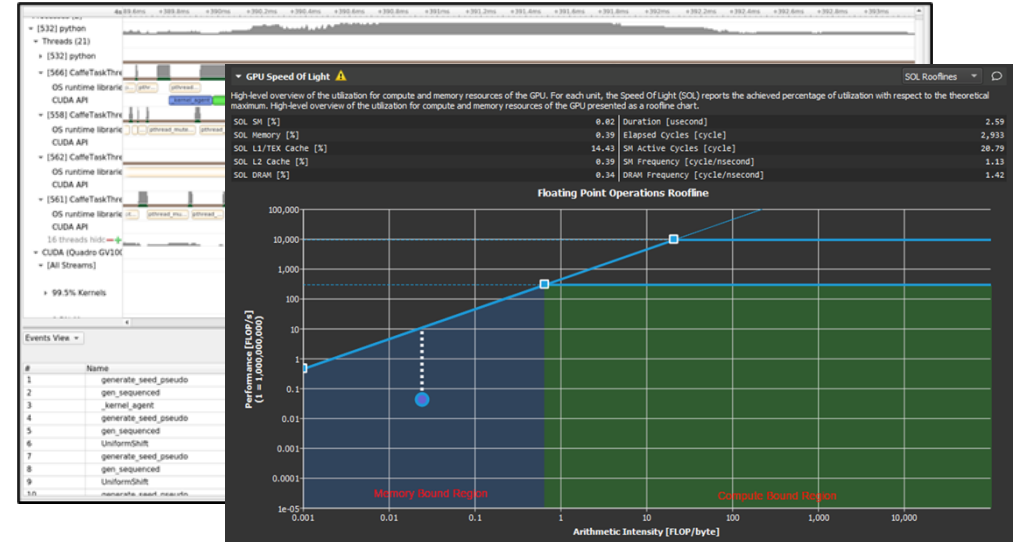
Name	Priority	Description
<input checked="" type="checkbox"/> GPU Speed Of Light Throughput (1)	10	High-level overview of the throughput for compu...
<input checked="" type="checkbox"/> GPU Speed Of Light Roofline Chart (1)	11	High-level overview of the utilization for comput...
<input checked="" type="checkbox"/> GPU Speed Of Light Hierarchical Roofline Chart (Double Precision)	12	High-level overview of the utilization for comp...
<input checked="" type="checkbox"/> GPU Speed Of Light Hierarchical Roofline Chart (Half Precision)	12	High-level overview of the utilization for comput...
<input checked="" type="checkbox"/> GPU Speed Of Light Hierarchical Roofline Chart (Single Precision)	12	High-level overview of the utilization for comput...
<input checked="" type="checkbox"/> GPU Speed Of Light Hierarchical Roofline Chart (Tensor Core)	12	High-level overview of the utilization for comput...
<input type="checkbox"/> Compute Workload Analysis (2)	20	Detailed analysis of the compute resources of t...

# Developer Tools

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



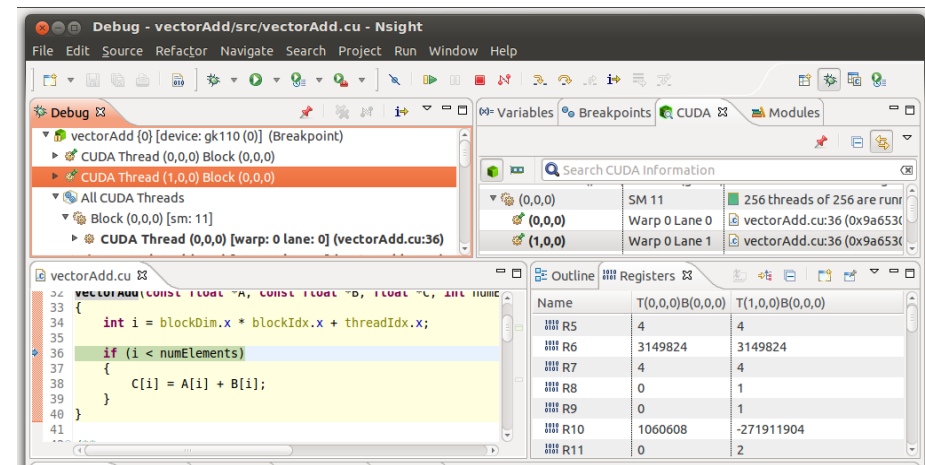
**Profilers:** Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)



**Correctness Checker:** Compute Sanitizer

```
$ compute-sanitizer --leak-check full memcheck_demo
===== COMPUTE-SANITIZER
Mallocing memory
Running unaligned_kernel
Ran unaligned_kernel: no error
Sync: no error
Running out_of_bounds_kernel
Ran out_of_bounds_kernel: no error
Sync: no error
===== Invalid __global__ write of size 4 bytes
===== at 0x60 in memcheck_demo.cu:6:unaligned_kernel(void)
===== by thread (0,0,0) in block (0,0,0)
===== Address 0x400100001 is misaligned
```

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





# Compute Debuggers

## Debug GPU kernels running on device

### ■ CUDA GDB

- CPU + GPU CUDA kernel debugger
- Supports stepping, breakpoints, in-line functions, variable inspection etc...
- Built on GDB and uses many of the same CLI commands
- Local/Remote connection support

### ■ Nsight Visual Studio Edition

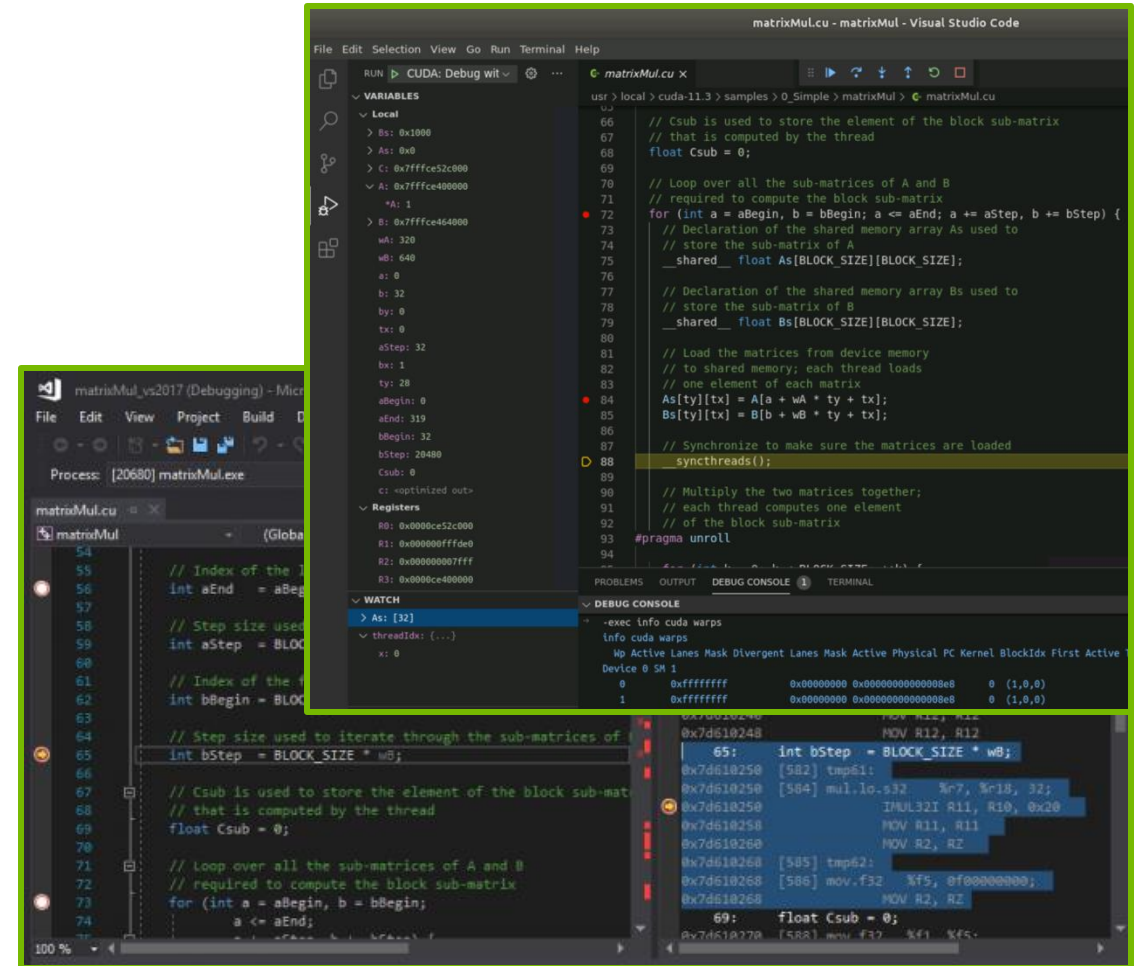
- IDE integration for Visual Studio
- Build and Debug CPU+GPU code from Visual Studio

### ■ Nsight Visual Studio Code Edition

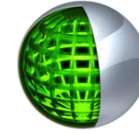
- New IDE integration for VS Code
- Build and Debug CPU+GPU code from Visual Studio Code
- Remotely target Linux targets from Windows or Linux

### ■ Nsight Eclipse Edition

- IDE integration for Eclipse
- Build and Debug CPU+GPU code from Eclipse

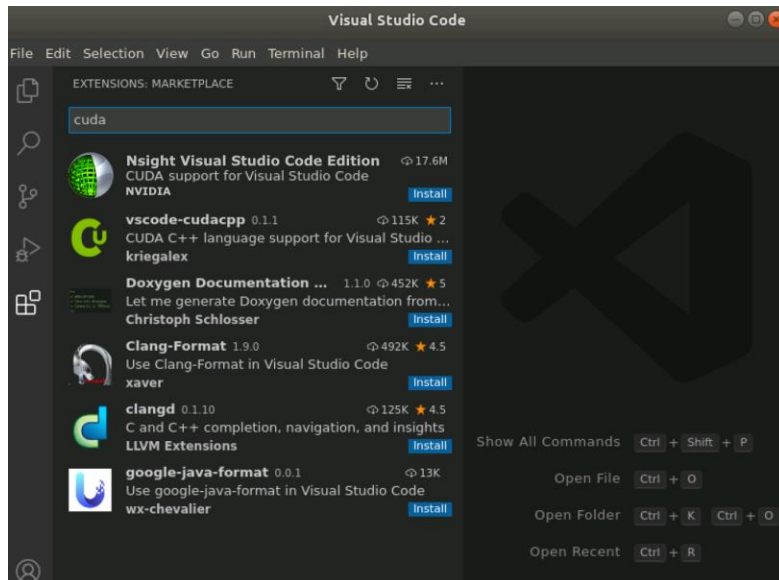


# Nsight Visual Studio Code Edition



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!



**VARIABLES**

```
Local
  > Bs: 0x1090
  > As: 0x0
  > C: 0x7ffcc52c000
  > A: 0x7ffcc400000
    *A: 1
  > B: 0x7ffcc464000
    wA: 320
    wB: 640
    a: 0
    b: 32
    by: 0
    tx: 0
    aStep: 32
    bx: 1
    ty: 28
    aBegin: 0
    aEnd: 319
    bBegin: 32
    bStep: 20480
    Csub: 0
    c: <optimized out>
```

**Registers**

```
RR: 0x0000ce52c000
R1: 0x000000ffde0
R2: 0x000000007fff
R3: 0x0000ce400000
```

**WATCH**

```
> As: [32]
  threadIdx: {...}
  x: 0
```

**BREAKPOINTS**

```
matrixMul.cu 72
matrixMul.cu /usr/local/cuda-... 61
matrixMul.cu /usr/local/cuda-... 64
matrixMul.cu /usr/local/cuda-... 72
```

**DEBUG CONSOLE**

```
-exec info cuda warps
Info cuda warps
Wp Active Lanes Mask Divergent Lanes Mask Active Physical PC Kernel BlockIdx First Active ThreadIdx
Device 0 SM 1
0 0xffffffff 0x00000000 0x00000000000000e8 0 (1,0,0) (0,0,0)
1 0xffffffff 0x00000000 0x00000000000000e8 0 (1,0,0) (0,1,0)
2 0xffffffff 0x00000000 0x00000000000000e8 0 (1,0,0) (0,2,0)
3 0xffffffff 0x00000000 0x00000000000000e8 0 (1,0,0) (0,3,0)
4 0xffffffff 0x00000000 0x00000000000000e8 0 (1,0,0) (0,4,0)
5 0xffffffff 0x00000000 0x00000000000000e8 0 (1,0,0) (0,5,0)
6 0xffffffff 0x00000000 0x00000000000000e8 0 (1,0,0) (0,6,0)
7 0xffffffff 0x00000000 0x00000000000000e8 0 (1,0,0) (0,7,0)
```

**CALL STACK**

```
(CUDA) PAUSED ON STEP
MatrixMulCUDA-32- matrixMul.cu 88
> matrixMul PAUSED
> matrixMul PAUSED
> cuda-EvtHandr PAUSED
> matrixMul PAUSED
```

**Status Bar:** Ln 88, Col 1 Spaces: 2 UTF-8 LF CUDA C++ CUDA: sm 1 warp 28 lane 0

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

# Compute Sanitizer

Automatically Scan for Bugs and Memory Issues

- Compute Sanitizer checks correctness issues via sub-tools:
  - **Memcheck** – Memory access error and leak detection tool.
  - **Racecheck** – Shared memory data access hazard detection tool.
  - **Initcheck** – Uninitialized device global memory access detection tool.
  - **Synccheck** – Thread synchronization hazard detection tool.

<https://github.com/NVIDIA/compute-sanitizer-samples>

```
$ make run_memcheck
/usr/local/cuda/compute-sanitizer/compute-sanitizer --destroy-on-device-error kernel memcheck_demo
===== COMPUTE-SANITIZER
Mallocing memory
===== Invalid __global__ write of size 4 bytes
=====   at 0x70 in unaligned_kernel()
=====   by thread (0,0,0) in block (0,0,0)
=====   Address 0x7f671ac00001 is misaligned
=====   and is inside the nearest allocation at 0x7fb654c00000 of size 4 bytes
=====   Saved host backtrace up to driver entry point at kernel launch time
=====   Host Frame: [0x2774ec]
=====           in /lib/x86_64-linux-gnu/libcuda.so.1
=====   Host Frame: __cudart803 [0xfccb]
=====           in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: cudaLaunchKernel [0x6a578]
=====           in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: cudaError_cudaLaunchKernel<char>(char const*, dim3, dim3, void**, unsigned
=====           in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: __device_stub_Z16unaligned_kernelv() [0xb22e]
=====           in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: unaligned_kernel() [0xb28c]
=====           in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: run_unaligned() [0xaf55]
=====           in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: main [0xb0e2]
=====           in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: ../sysdeps/nptl/libc_start_call_main.h:58: __libc_start_call_main [0x2dfd0]
=====           in /lib/x86_64-linux-gnu/libc.so.6
```

# Compute Sanitizer

## Reading a Memcheck Example Report

```
===== Invalid __global__ write of size 4 bytes
===== at 0xb0
===== by thread
===== Address 0x87654320 is out of bounds
=====
===== Device and host backtraces
===== Device Frame:/home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo.cu:44:out_of_bounds_kernel() [0x30]
===== Saved host backtrace up to driver entry point at kernel launch time
===== Host Frame: [0x2774ec]
===== in /lib/x86_64-linux-gnu/libcuda.so.1
===== Host Frame:__cudart803 [0xfccb]
===== in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
===== Host Frame:cudaLaunchKernel [0x6a578]
===== in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
```

Address space      Type of access      Access size

Access location

Faulty thread

Faulty address and nearest allocation

Device and host backtraces

## Other Resources

Continue your learning journey. Keep engaged after the event. [www.openhackathons.org](http://www.openhackathons.org)



### [Open Hackathons GitHub](#)

Explore additional  
Bootcamp materials  
and resources



### [TACC Open Hackathon](#)

Deadline: August 7, 2024

- Advance and accelerate your science
- Work with dedicated mentors
- Access the latest systems



### [NVIDIA DLI](#)

Resources for diverse  
training  
Explore AI, accelerated  
computing, data science,  
graphics, and more.

# Useful Links

Web: <https://developer.nvidia.com/tools-overview>

How to contact us?

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

email: [devtools-support@nvidia.com](mailto:devtools-support@nvidia.com)

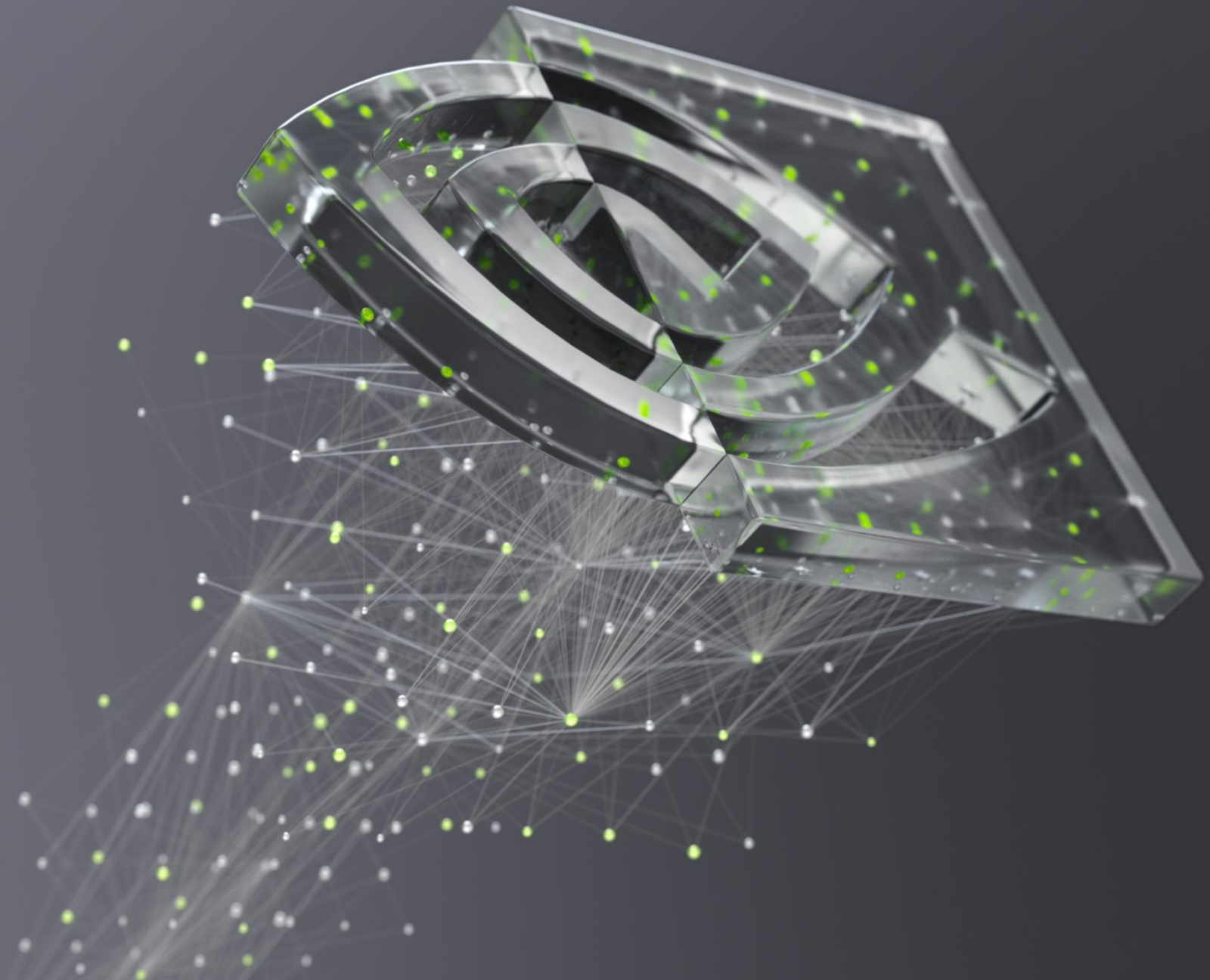
Other digital GTC talks of interest:

[S21351](#): Scaling the Transformer Model Implementation in PyTorch Across Multiple Nodes

[S21547](#): Rebalancing the Load: Profile-Guided Optimization of the NAMD Molecular Dynamics Program for Modern GPUs using Nsight Systems

S21771: Optimizing CUDA Kernels in HPC Simulation and Visualization Codes using Nsight Compute

[S21565](#): Roofline Performance Model for HPC and Deep-Learning Applications



**nVIDIA**<sup>®</sup>