NVIDIA Developer Tools

Kristopher Keipert, Solutions Architect

ATPESC 2015

kkeipert@nvidia.com

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

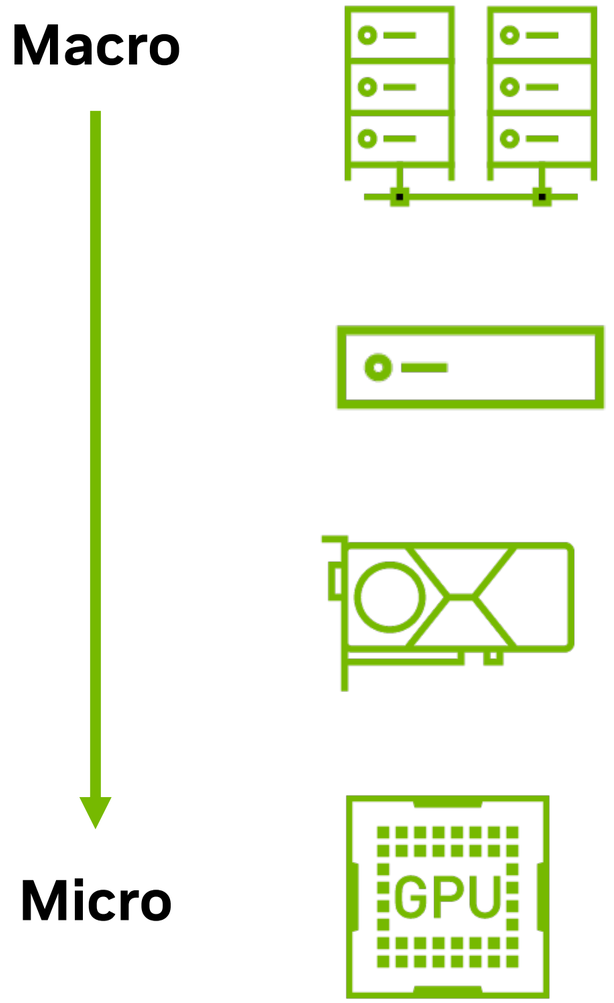
```
__global__
void saxpy(int n, float a,
            float *x, float *y) {
  int i = blockIdx.x*blockDim.x +
          threadIdx.x;
  if (i < n) y[i] += a*x[i];
}

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

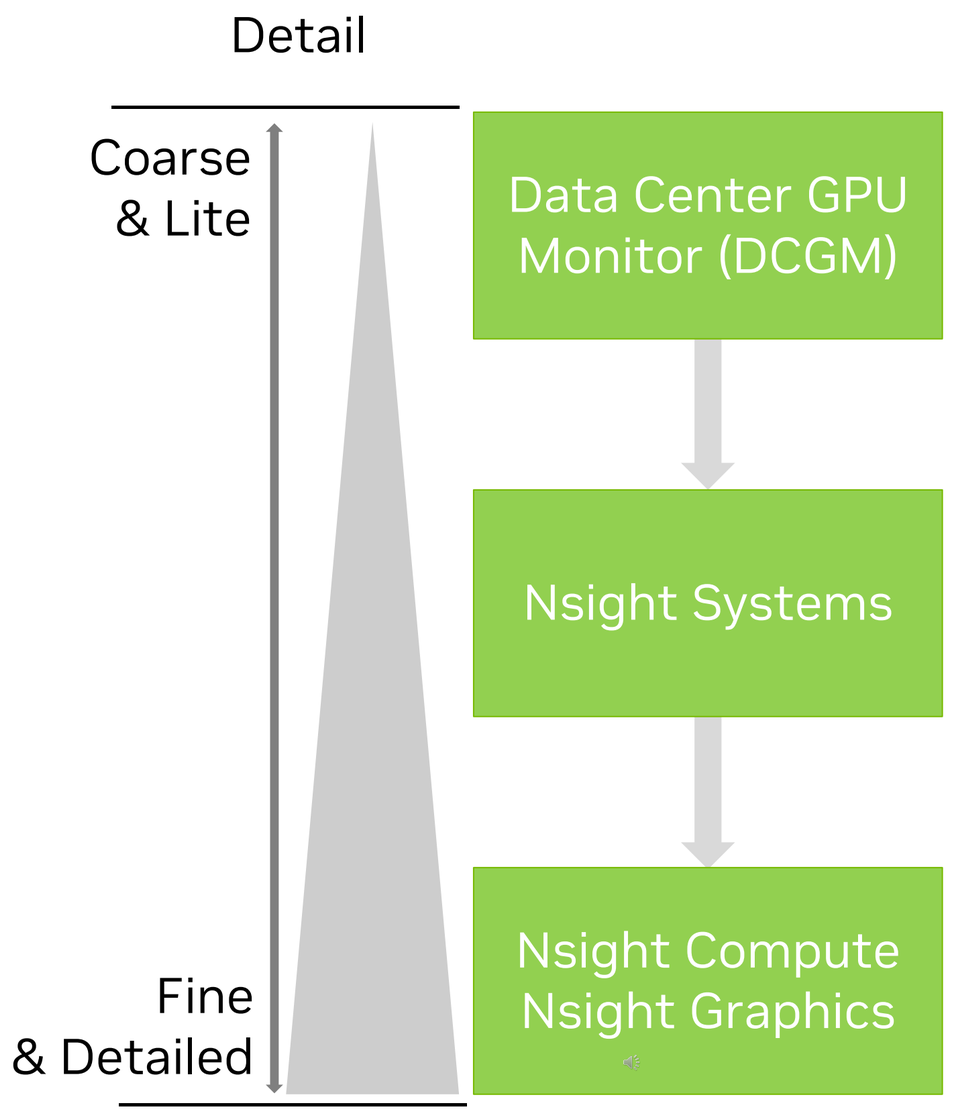
Agenda

- High-level tools ecosystem
- For each tool:
  - Brief description and feature overview
  - What's new in the latest releases
  - Focus on problem solving and application in the system hierarchy
- Summary - Putting it all together



Monitoring → Profiling

- Different target users, intent, details, design tradeoffs
  - Some overlap but not interchangeable
- Monitors
  - Goal: Coarsely observe quality, utilization, progress, help rate jobs
  - Users: Admins & users of top, netstats, taskman, …
  - Track issues back to jobs
    - Low rate (minutes or seconds)
    - Sensible to observer checking & reacting, low overhead
    - Less details towards root cause
  - Uptime: Usually 100%
- Profilers
  - Goal: Aid in application optimization
  - Users: Engineers looking to relating back to areas of code
  - Uptime: Usually not 100%



Data Center GPU Monitor (DCGM)

- Very unobtrusive
- Great for fleet health, job health, job rating
- Optimized for low frequency statistics (10hz or less)
- Maps to jobs & processes
  - But not code & algorithms, like profilers!
- Pause/resume to profile
  - CLI
    - dcgmi profile --pause
    - dcgmi profile --resume
  - API
    - dcgmProfPause
    - dcgmProfResume


```
(base)  avahuja@avahuja-ws  ~  dcgmi dmon -e 1002,1003,1004,1007,1005,1009,1010  -i 0
# Entity  SMACT  SMOCC  TENSO  FP32A  DRAMA                PCITX                PCIRX
     Id
    GPU 0  1.000  0.654  0.000  0.006  0.338            161979752             53993250
    GPU 0  0.949  0.558  0.036  0.009  0.415             27167886            184237298
    GPU 0  0.762  0.417  0.126  0.005  0.497              7823530             23181418
    GPU 0  0.762  0.417  0.126  0.005  0.497              7823530             23181418
    GPU 0  0.994  0.590  0.129  0.006  0.475             11681230             33786208
    GPU 0  0.996  0.630  0.014  0.007  0.438             12674795             35978628
    GPU 0  0.942  0.577  0.024  0.008  0.414             27839447            186253850
    GPU 0  0.797  0.426  0.105  0.006  0.479              7361822             21774288
    GPU 0  0.939  0.542  0.165  0.005  0.490             10775260             31687034
    GPU 0  0.995  0.631  0.014  0.007  0.389             12194498             35853104
    GPU 0  0.955  0.586  0.024  0.008  0.409             28487102            187685151
    GPU 0  0.936  0.513  0.106  0.008  0.576              8269131             24102604
    GPU 0  0.832  0.472  0.165  0.004  0.459             10113749             29940997
    GPU 0  0.996  0.637  0.007  0.007  0.373             13152539             38756920
    GPU 0  0.954  0.600  0.017  0.007  0.422             29765728            191608205

# Entity  SMACT  SMOCC  TENSO  FP32A  DRAMA                PCITX                PCIRX
     Id
    GPU 0  0.984  0.530  0.105  0.009  0.573              8164627             23735310
    GPU 0  0.779  0.431  0.182  0.004  0.465              9875826             28424670
    GPU 0  0.996  0.637  0.007  0.007  0.372             12673519             36793740
    GPU 0  0.993  0.630  0.013  0.006  0.409             13274947             37754470
    GPU 0  0.945  0.511  0.076  0.010  0.545             25196587            178239740
    GPU 0  0.726  0.380  0.220  0.003  0.494             18350732             55105939
    GPU 0  0.997  0.638  0.007  0.007  0.370             12122060             35676752
```

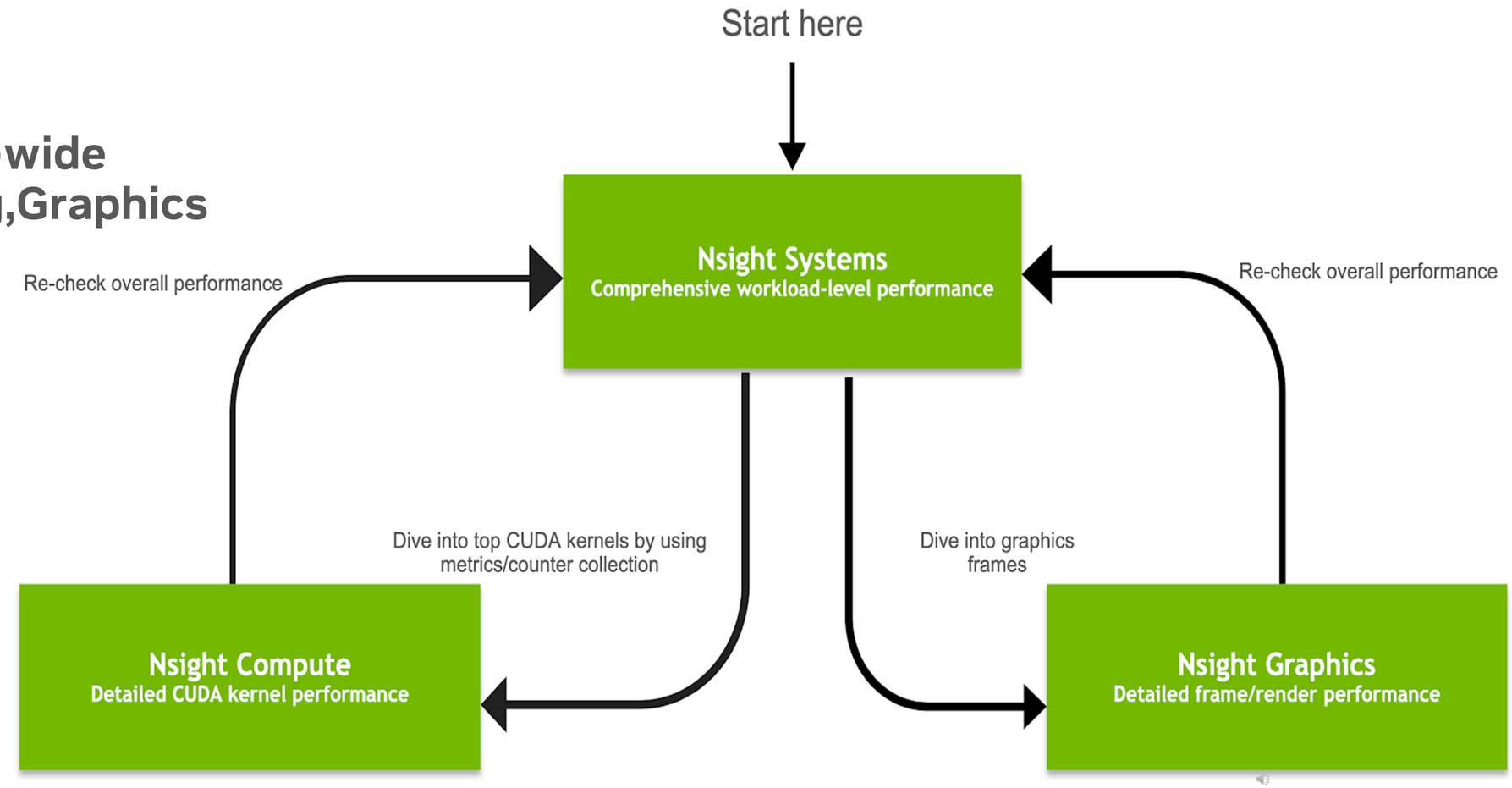
Nsight Profiler Family

Performance Analysis Workflow

Nsight Systems
Analyze algorithms system-wide
CPU,GPU,CUDA,Networking,Graphics

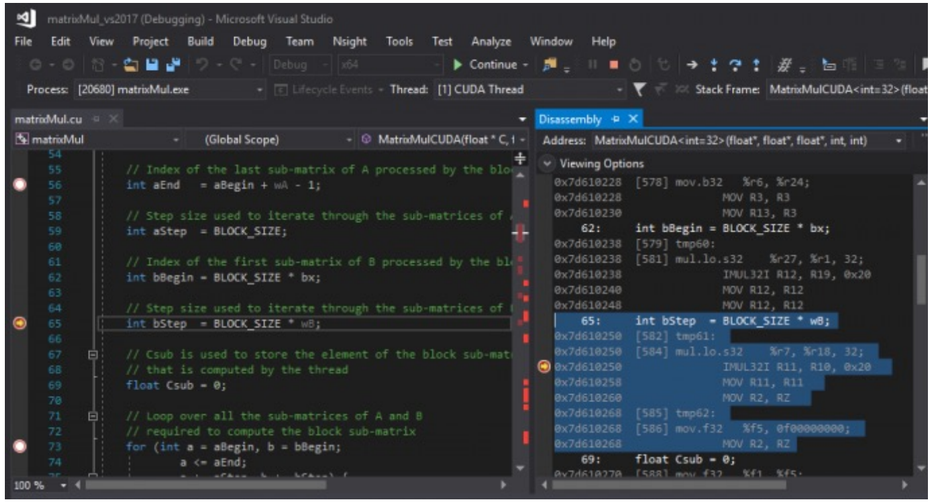
Nsight Compute
Debug & analyze
CUDA kernels

Nsight Graphics
Debug & analyze
Graphics shaders & frames



Developer Tools Ecosystem

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

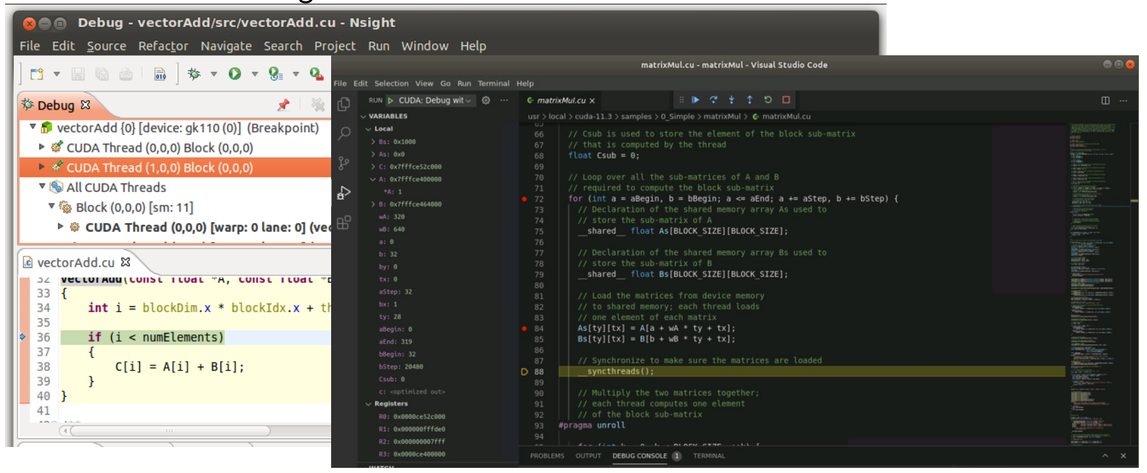


Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)

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 Eclipse Edition
Nsight Visual Studio Edition
Nsight Visual Studio **Code** Edition

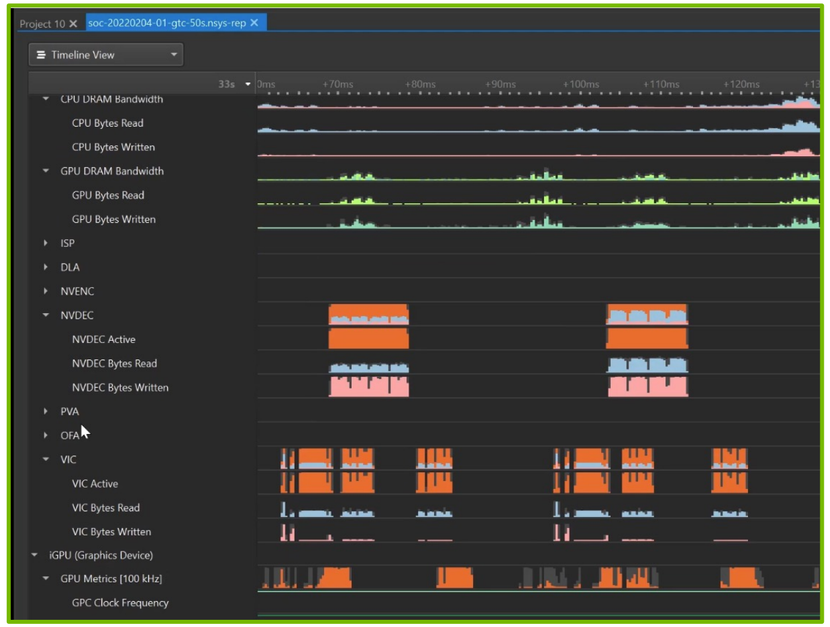


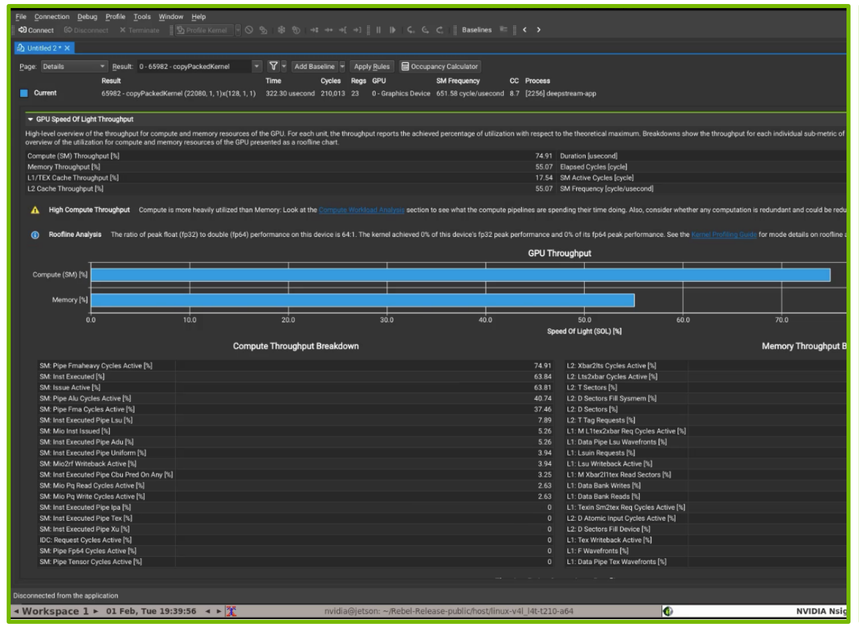
NVIDIA

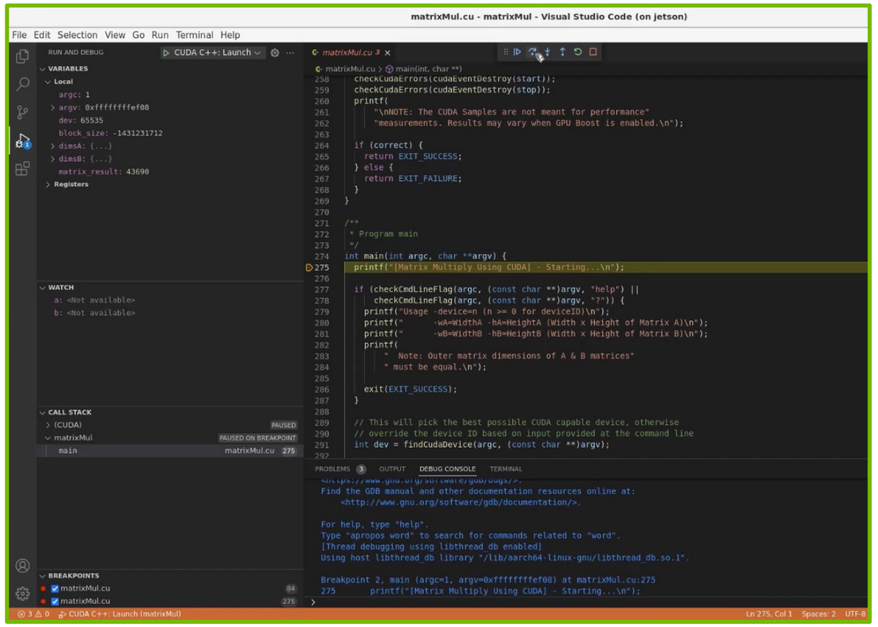
GUI Support for Grace and other Arm Platforms

Nsight Systems - Nsight Compute - Nsight Visual Studio Code Edition

- GUIs run natively on
  - NVIDIA® Jetson AGX Orin™ SoC
  - Arm Server Platforms
- Use new native GUI or existing remote collection capabilities







Updated Websites/Docs/Icons

NVIDIA Developer Tools

NVIDIA Nsight™ tools are a powerful set of libraries, SDKs, and developer tools spanning across desktop and mobile targets that enable developers to build, debug, profile, and develop software that utilizes the latest accelerated computing hardware.

	IDE / DEBUGGING						PROFILERS / DESIGN				LIBRARIES / API / SDK			
	Nsight Visual Studio Edition	Nsight Eclipse Edition	Nsight Visual Studio Code Edition	CUDA-GDB	Compute Sanitizer	Nsight Graphics	Nsight Compute	Nsight Systems	Nsight Deep Learning Designer	Nsight Graphics	CUPTI	NVTX	Nsight Aftermath SDK	Nsight Perf SDK
CUDA	●	●	●	●	●		●	●	●		●	●		
GRAPHICS						●		●	●	●		●	●	●
OPTIX	●	●	●	●	●		●	●			●	●		

What Segment Are You Working on?

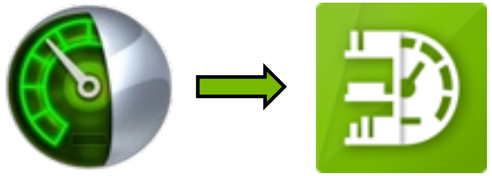
View All | CUDA/Compute | Graphics | OptiX | Deep Learning

Developer Activity

CODE DEVELOPMENT	DEBUGGING / CORRECTNESS	PROFILING
PLATFORM ANALYSIS	KERNEL ANALYSIS	APIs

Nsight Compute

Nsight Compute is an interactive kernel profiler for CUDA applications. It provides detailed performance metrics and API



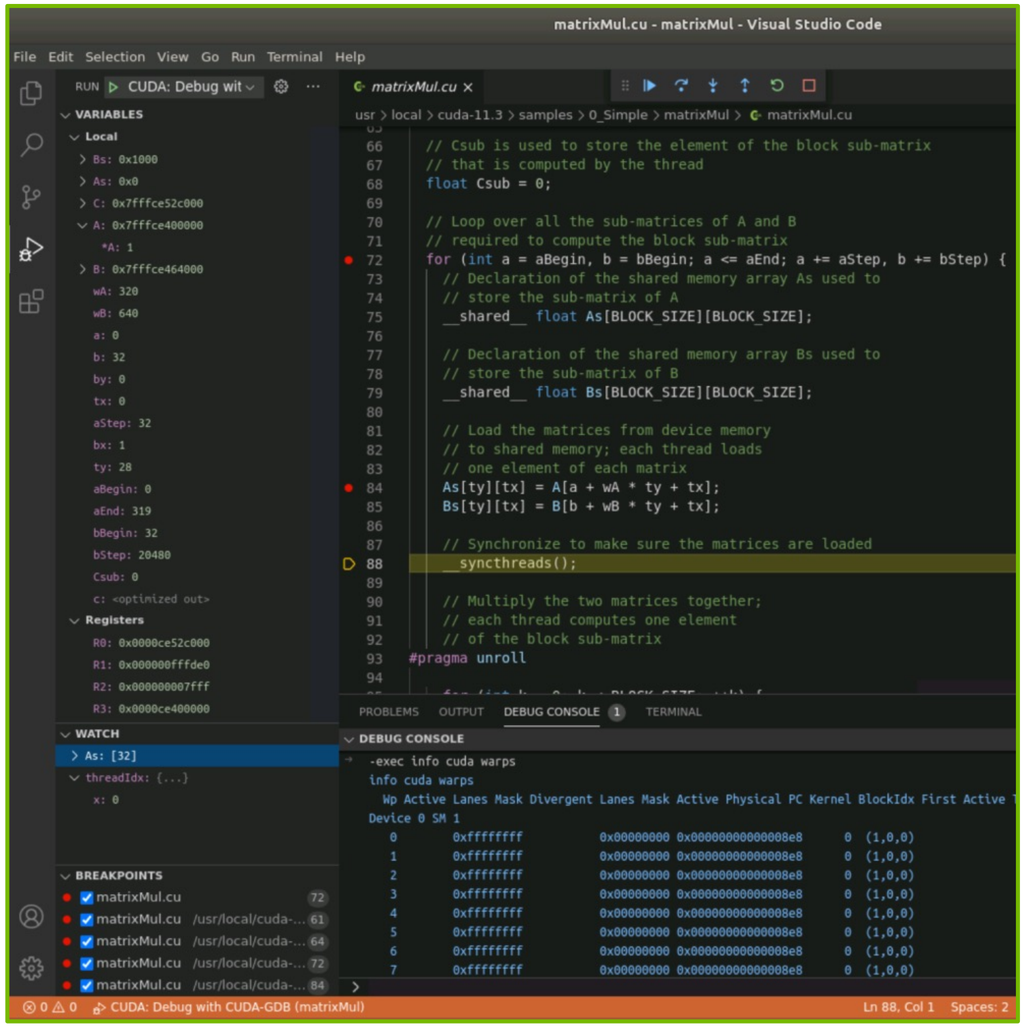
https://developer.nvidia.com/tools-overview

Compute Debuggers/IDEs

Compute Debuggers

Debug GPU Kernels Running on Device

- CUDA GDB
  - CPU + GPU CUDA kernel debugger
  - Supports stepping, breakpoints, in-line functions, variable inspection etc…
- 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



CUDA GDB

Command-line and IDE Back-end Debugger

- Unified CPU and CUDA Debugging
- CUDA-C/SASS support
- Built on GDB and uses many of the same CLI commands
- Local/Remote connection support
- Backend for IDE debuggers

```
(cuda-gdb) info cuda threads breakpoint all
  BlockIdx ThreadIdx         Virtual PC Dev SM Wp Ln         Filename  Line
Kernel 0
   (1,0,0)    (0,0,0) 0x0000000000948e58   0 11  0  0 infoCommands.cu    12
   (1,0,0)    (1,0,0) 0x0000000000948e58   0 11  0  1 infoCommands.cu    12
   (1,0,0)    (2,0,0) 0x0000000000948e58   0 11  0  2 infoCommands.cu    12
   (1,0,0)    (3,0,0) 0x0000000000948e58   0 11  0  3 infoCommands.cu    12
   (1,0,0)    (4,0,0) 0x0000000000948e58   0 11  0  4 infoCommands.cu    12
   (1,0,0)    (5,0,0) 0x0000000000948e58   0 11  0  5 infoCommands.cu    12

(cuda-gdb) info cuda threads breakpoint 2 lane 1
  BlockIdx ThreadIdx         Virtual PC Dev SM Wp Ln         Filename  Line
Kernel 0
   (1,0,0)    (1,0,0) 0x0000000000948e58   0 11  0  1 infoCommands.cu    12
```

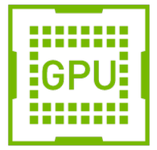
```
117  /**
118   * Run a simple test of matrix multiplication using CUDA
119   */
120  int MatrixMultiply(int argc, char **argv, int block_size, const dim3 &dimsA,
121                     const dim3 &dimsB) {
122    // Allocate host memory for matrices A and B
123    unsigned int size_A = dimsA.x * dimsA.y;
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                    Filter (e.g. text, !exclude)

96          Csub += As[ty][k] * Bs[k][tx];
cuda sm 0 warp 1 lane 0
CUDA focus unchanged.

Thread 1 "matrixMul" hit Breakpoint 2, MatrixMulCUDA<32><<<(20,10,1),(32,32,1)>>> (C=0x7fffccb2c000, A=0x7fffcca00000, B=0x7fffcca64000, wA=320, wB=640) at matrixMul.cu:96
96          Csub += As[ty][k] * Bs[k][tx];

Thread 1 "matrixMul" hit Breakpoint 3, MatrixMulCUDA<32><<<(20,10,1),(32,32,1)>>> (C=0x7fffccb2c000, A=0x7fffcca00000, B=0x7fffcca64000, wA=320, wB=640) at matrixMul.cu:108
108       C[c + wB * ty + tx] = Csub;
> |
```

```
⊗ 0 ⚠ 0   CUDA: Debug with CUDA-GDB (matrixMul)   Ln 107, Col 1  Spaces: 2  UTF-8  LF  CUDA C++
```

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.

```
$ 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
```

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

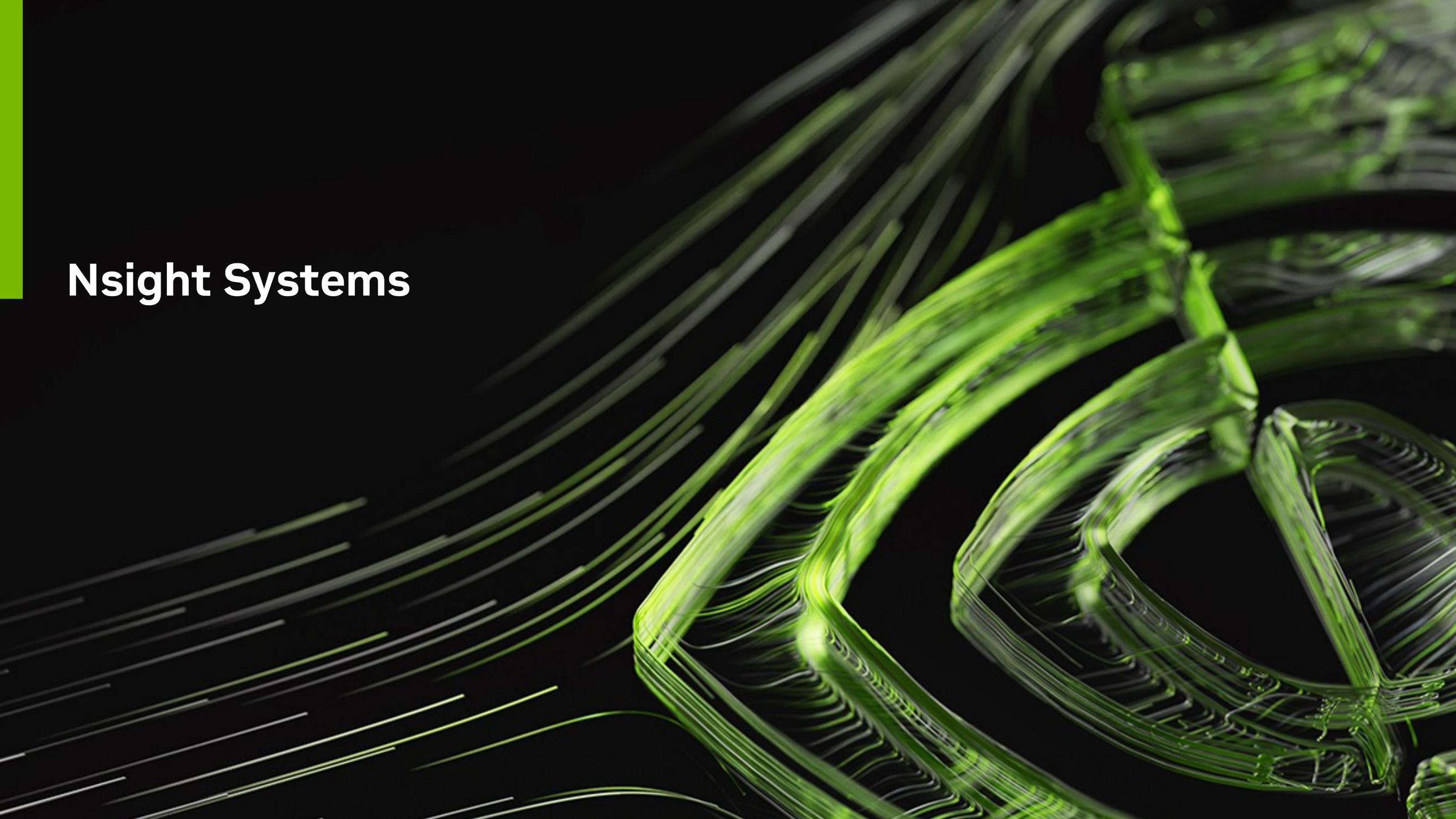
Compute Debuggers/IDEs
New Features

Correctness Tools Features

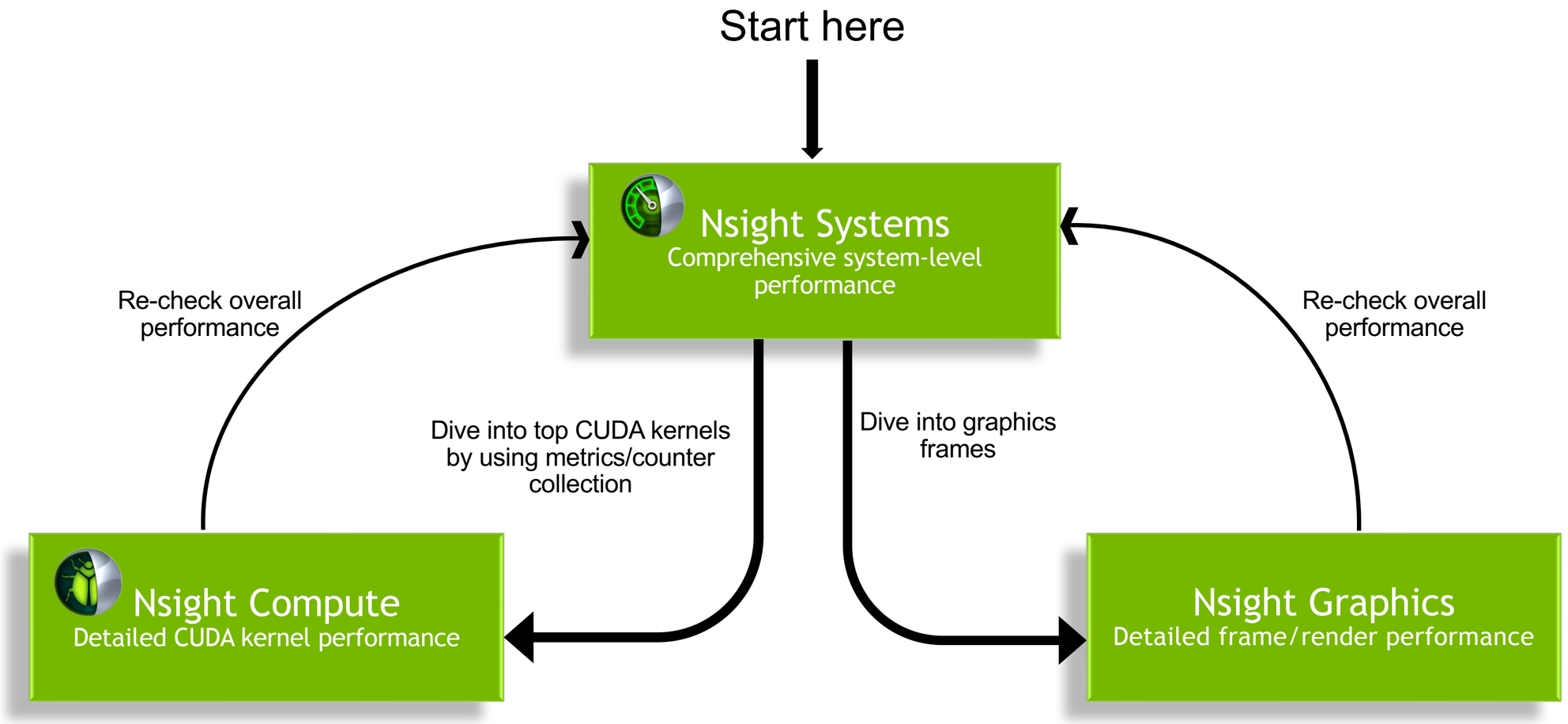
- Debuggers using **Unified Backend**
  - Consistent debugging experience across platforms
  - Performance and usability improvements
  - New features “float all boats”
- New Compute Sanitizer use cases
  - Racecheck support for async-copy, DSMEM (Hopper), and GMMA (Hopper)
  - Memcheck for cache control operations
  - Improved core dump support
  - Multiple OptiX analysis improvements
- Nsight Visual Studio Code Edition
  - New support for debugging containers

```
========= COMPUTE-SANITIZER
========= Invalid __global__ write of size 1 bytes
=========     at 0x4d70 in
/home/cuda/optixBasic/draw_solid_color.cu:69:__raygen__draw_solid_co
lor_0xebf766b2f0642d4e
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x7f878f900403 is out of bounds
=========     and is 262,132 bytes after the nearest allocation at
0x7f878f8c0400 of size 16 bytes
=========     Device Frame:NVIDIA internal [0x430]
=========     Saved host backtrace up to driver entry point at
kernel launch time
=========     Host Frame: [0x60fbaa]
=========                 in /lib/x86_64-linux-gnu/libnvoptix.so.1
=========     Host Frame:optix_stubs.h:568:optixLaunch [0xe1ff]
=========                 in /home/cuda/optixBasic/optixBasic
=========     Host
Frame:/home/cuda/optixBasic/optixBasic.cpp:227:main [0xb735]
=========                 in /home/cuda/optixBasic/optixBasic
=========     Host
Frame:../sysdeps/nptl/libc_start_call_main.h:58:__libc_start_call_ma
in [0x2dfd0]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame:../csu/libc-start.c:379:__libc_start_main
[0x2e07d]
=========                 in /lib/x86_64-linux-gnu/libc.so.6
=========     Host Frame: [0x8dde]
=========                 in /home/cuda/optixBasic/optixBasic
```

Nsight Systems



NSIGHT PROFILERS WORKFLOW



Nsight Systems

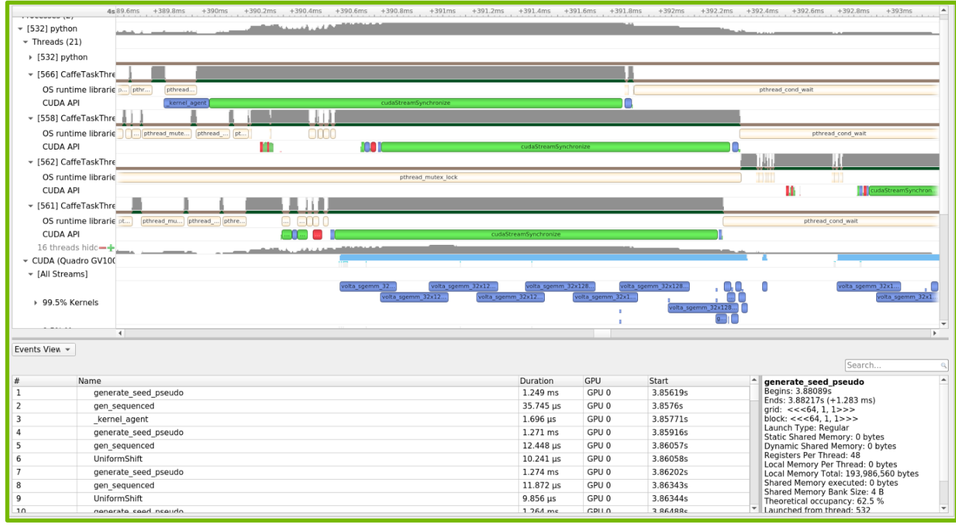
System Profiler

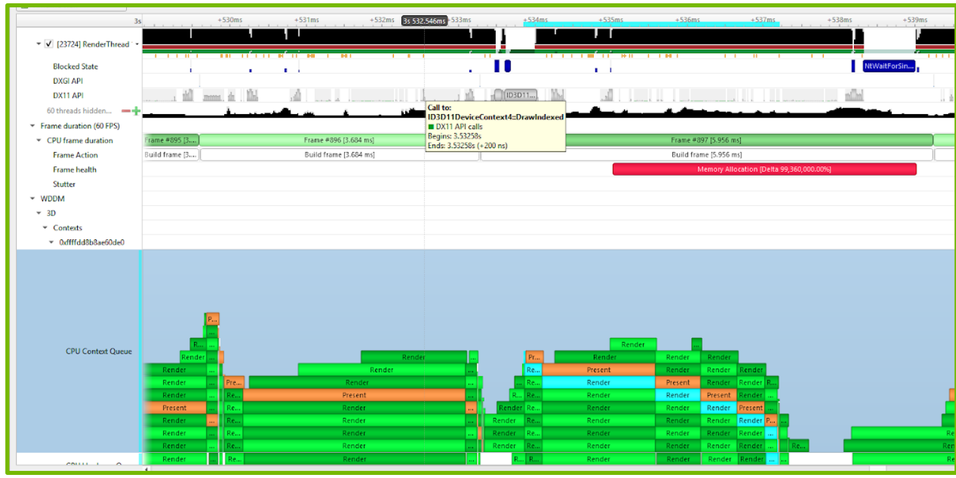
Key Features:

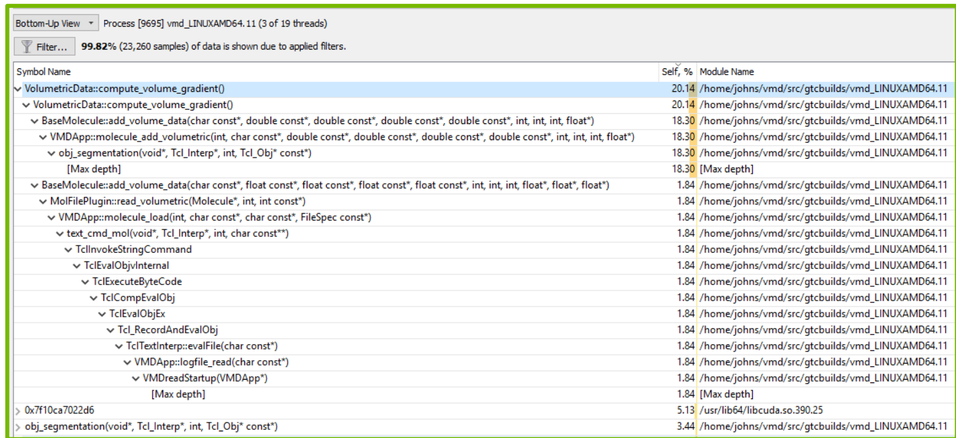
- System-wide application algorithm tuning
  - Multi-process tree support
- Locate optimization opportunities
  - Visualize millions of events on a very fast GUI timeline
  - Identify gaps of unused CPU and GPU time
- Balance your workload across multiple CPUs and GPUs
  - CPU algorithms, utilization and thread state
    GPU streams, kernels, memory transfers, etc
- Command Line, Standalone, IDE Integration

- OS: Linux (x86, Power, ARM Server, Tegra), Windows, macOS X (host)
- GPUs: Pascal+

- Docs/product: https://developer.nvidia.com/nsight-systems







CPU cores

GPU metrics samples

SMs active

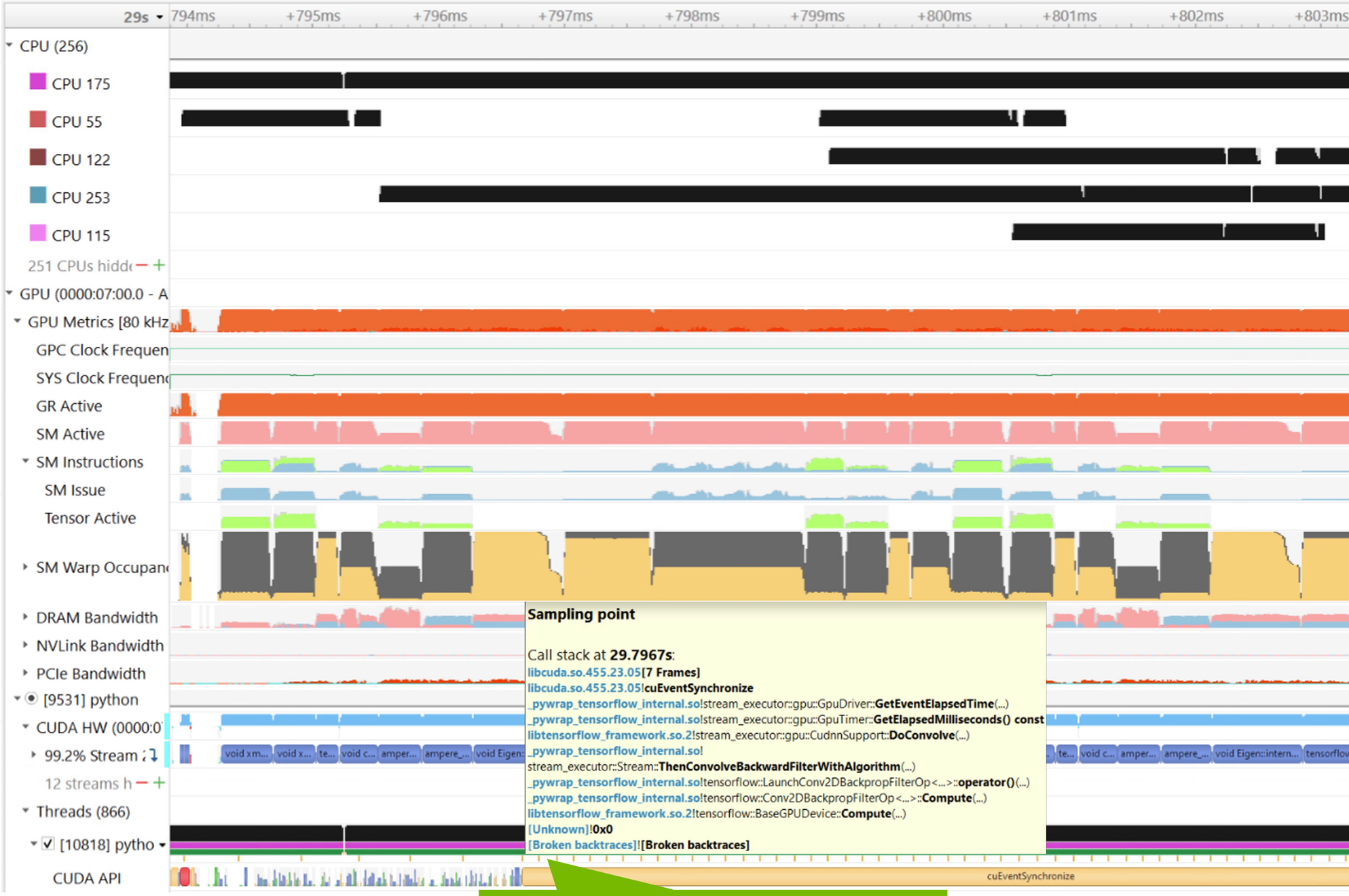
SM instructions TensorCores

GPU bandwidths

GPU CUDA kernel & memory activities

CPU processes & thread states

CUDA API trace



CPU call-stack samples

NVIDIA

Zoom/Filter to Exact Areas of Interest

Pixel Time Coverage Based Level Of Detail (LOD)

Zooming in reveals gaps where there were valleys

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
```

Core Strategy

- What's HOT?
  - Will it be easier to shrink what I coded?
  - This is where MOST people concentrate. …intuitive but not always best !/$
- What's COLD?
  - Will it be easier to take advantage of the something unused?
  - Free money? Yes please!

- Hot spots might be:
  - Parallelizable?
  - Shrinkable without compromising accuracy, memory, etc
- Cold spots are clear, measurable opportunities!!!
  - How can i remove or fill them?
  - Where do I have incorrect/unnecessary/unexpected dependencies & synchronization?
    - Between threads, processes or across nodes (ex: MPI_Barrier)!

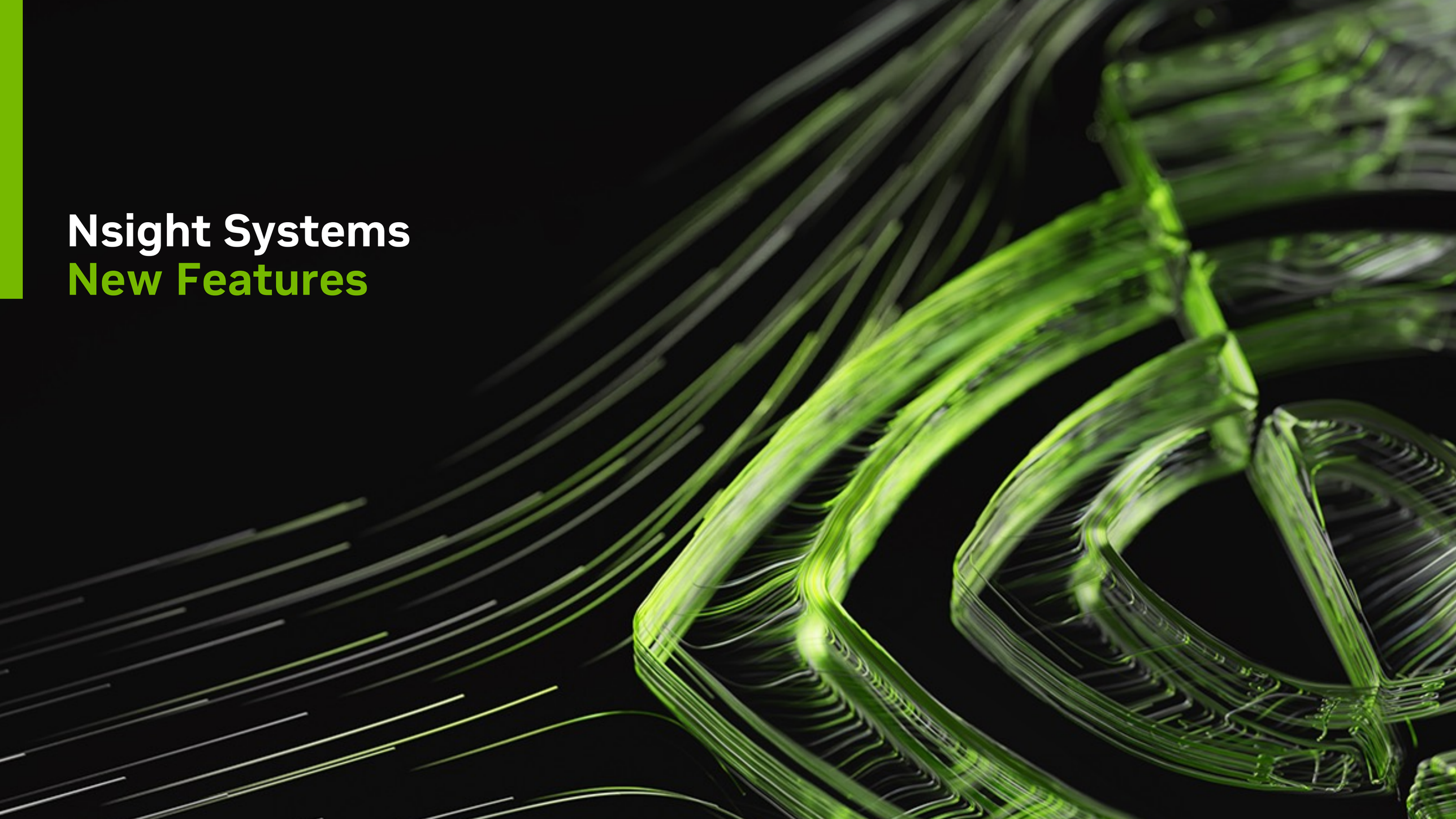
General optimization tips

- Using tensor cores?
- Increase grid and batch size to utilize GPUs width
- Conventional parallelism – more worker threads!
- Parallel pipelining
  - No data dependency? Parallelize!
  - Prefetch next batch/iteration during computation

General optimization tips

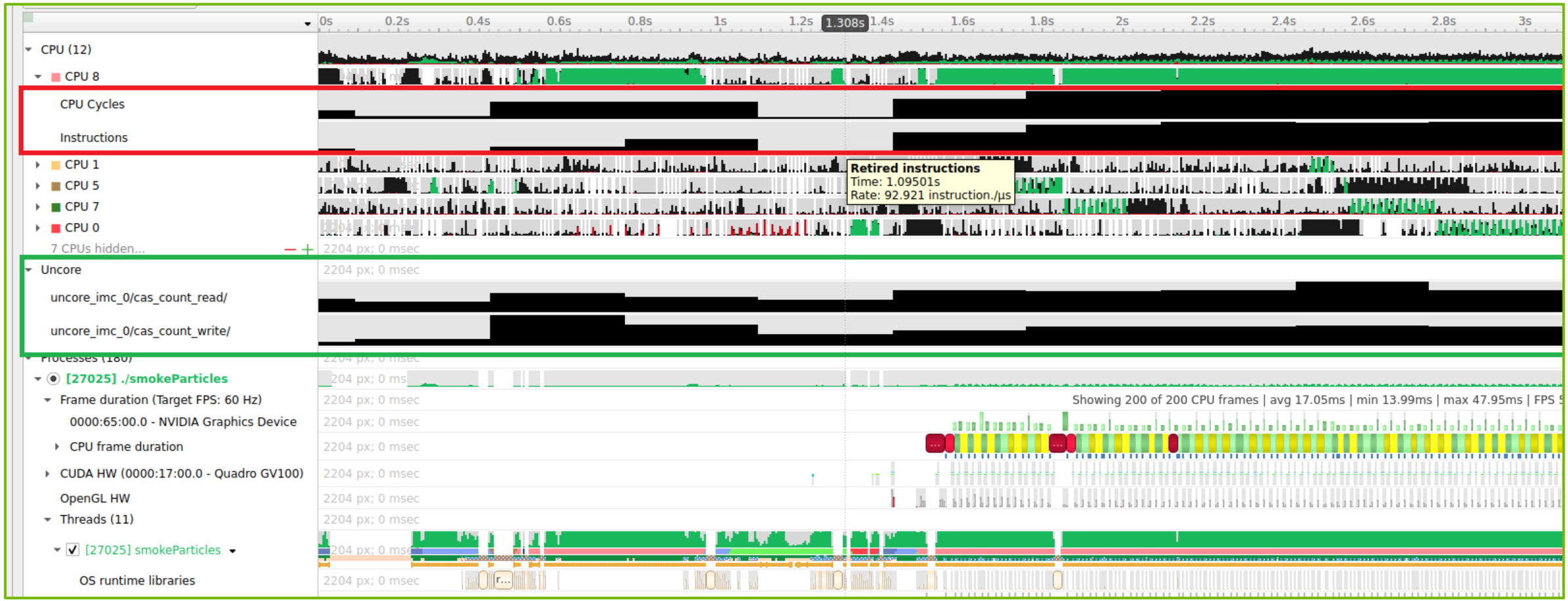
- Fuse tiny kernels, copies, memsets.
  - Check out CUDA Graphs
- Overlap/oversubscribe with MPS
- Multi-buffering
  - Don’t make everyone wait on the same piece of memory
  - Double, triple buffer
- Avoid moving data back to the CPU
  - Pre-allocate and recycle!
- Minimize managed memory page faults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

Nsight Systems
New Features



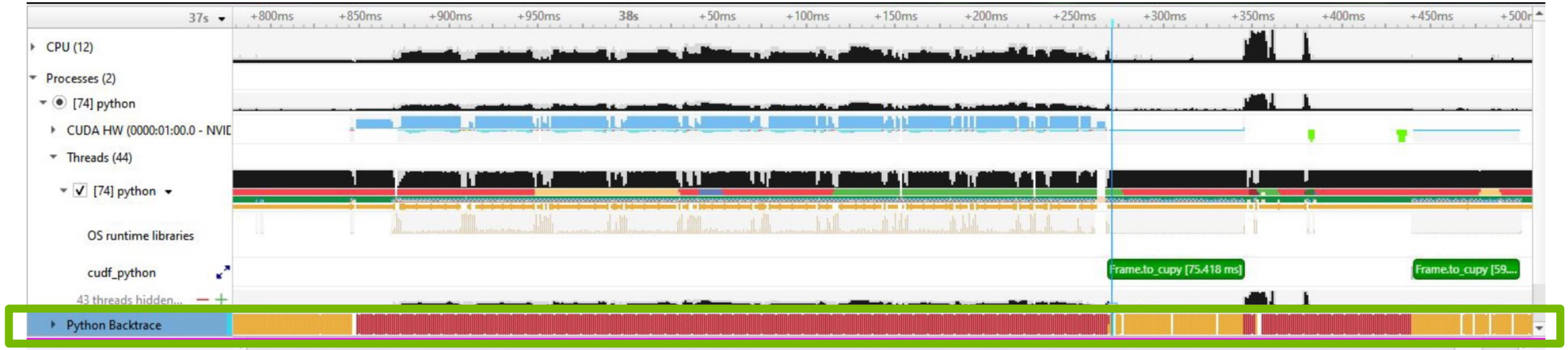
CPU Counters and OS Metrics

- CPU X86_64/SBSA hardware counters
  - Cache misses, DRAM access, etc…



Python Call Stack Sampling

- Periodically sampled stacks
- Correlated with all other collected activity



Events View

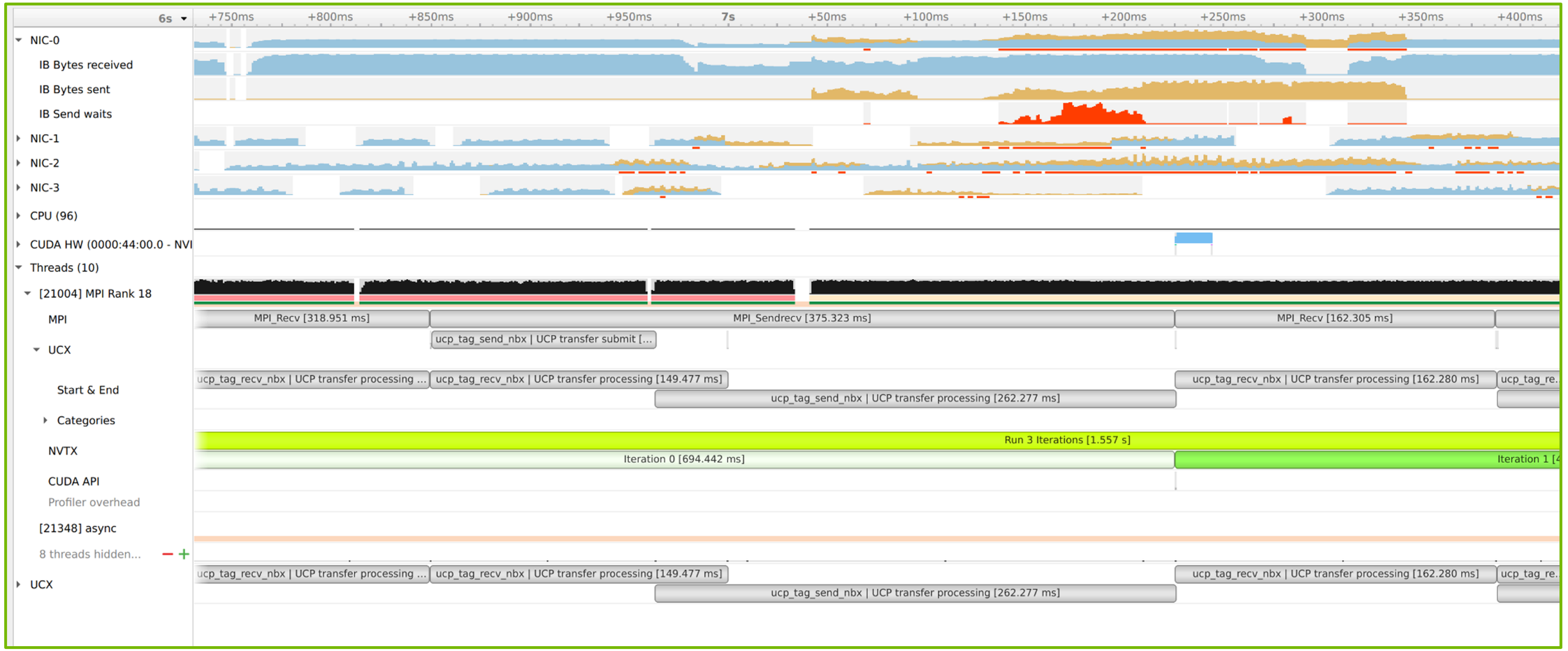
#	Name	Start	TID	Category
33352		38.2721s	82	Python Thread [1]
33353		38.2734s	82	Python Thread [1]
33354		38.2746s	82	Python Thread [1]
33355		38.2757s	82	Python Thread [1]
33356		38.2768s	82	Python Thread [1]
33357		38.278s	82	Python Thread [1]
33358		38.2792s	82	Python Thread [1]
33359		38.2805s	82	Python Thread [1]
33360		38.2816s	82	Python Thread [1]
33361		38.2828s	82	Python Thread [1]
33362		38.284s	82	Python Thread [1]
33363		38.2853s	82	Python Thread [1]
33364		38.2864s	82	Python Thread [1]
33365		38.2878s	82	Python Thread [1]

Description:

Time: 38.2721s
Thread: 82
CategoryId: 1
Category: Python Thread
PythonBacktrace
Thread state: Running
Frames count: 20
Frames (Function, File, Line)
0: safe_cuda_api_call, /opt/conda/envs/rapids/lib/python3.9/site-packages/numba/cuda/cudadrv/driver.py, 319
1: __enter__, /opt/conda/envs/rapids/lib/python3.9/site-packages/numba/cuda/cudadrv/driver.py, 498
2: ensure_context, /opt/conda/envs/rapids/lib/python3.9/site-packages/numba/cuda/cudadrv/devices.py, 121
3: __enter__, /opt/conda/envs/rapids/lib/python3.9/contextlib.py, 119
4: _require_cuda_context, /opt/conda/envs/rapids/lib/python3.9/site-packages/numba/cuda/cudadrv/devices.py, 231
5: as_cuda_array, /opt/conda/envs/rapids/lib/python3.9/site-packages/numba/cuda/api.py, 76
6: data_array_view, /opt/conda/envs/rapids/lib/python3.9/site-packages/cudf/core/column/column.py, 119
7: values, /opt/conda/envs/rapids/lib/python3.9/site-packages/cudf/core/column/column.py, 177
8: , /opt/conda/envs/rapids/lib/python3.9/site-packages/cudf/core/frame.py, 536
9: get_column_values_na, /opt/conda/envs/rapids/lib/python3.9/site-packages/cudf/core/frame.py, 477
10: _to_array, /opt/conda/envs/rapids/lib/python3.9/site-packages/cudf/core/frame.py, 498
11: to_cupy, /opt/conda/envs/rapids/lib/python3.9/site-packages/cudf/core/frame.py, 533
12: inner, /opt/conda/envs/rapids/lib/python3.9/contextlib.py, 79
13: input_to_cuml_array, /opt/conda/envs/rapids/lib/python3.9/site-packages/cuml/common/input_utils.py, 338

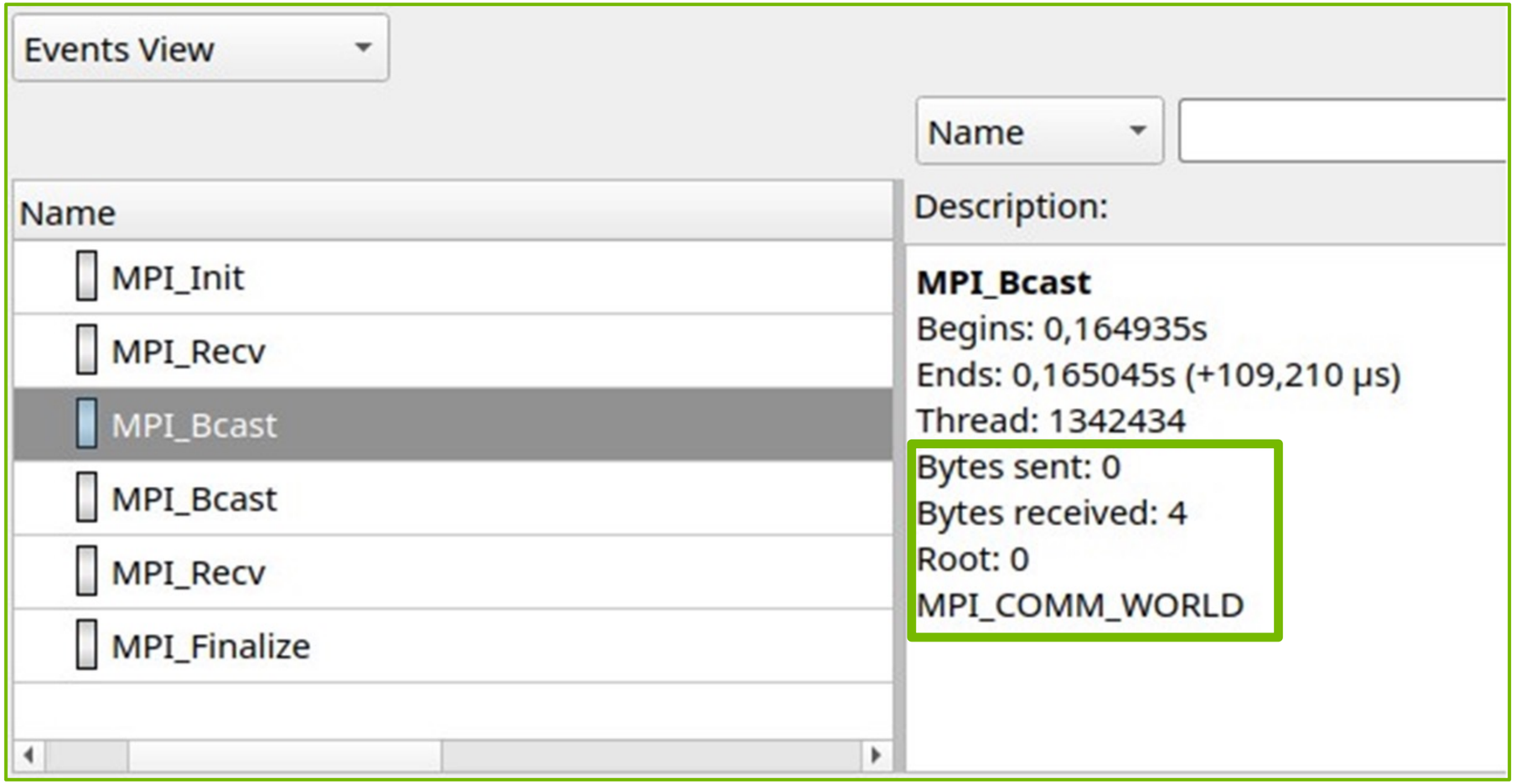
NVIDIA Networking NIC/DPU Metrics Sampling

- Includes InfiniBand and Ethernet
- Sent / Received / Waits
- Correlate with expected network traffic and other system activities



Network API Trace - Extended Descriptions/Tooltips

- Supports MPI versions
  - OpenMPI
  - MPICH
- Data for each API call
- Identify performance trends and specific underperforming calls

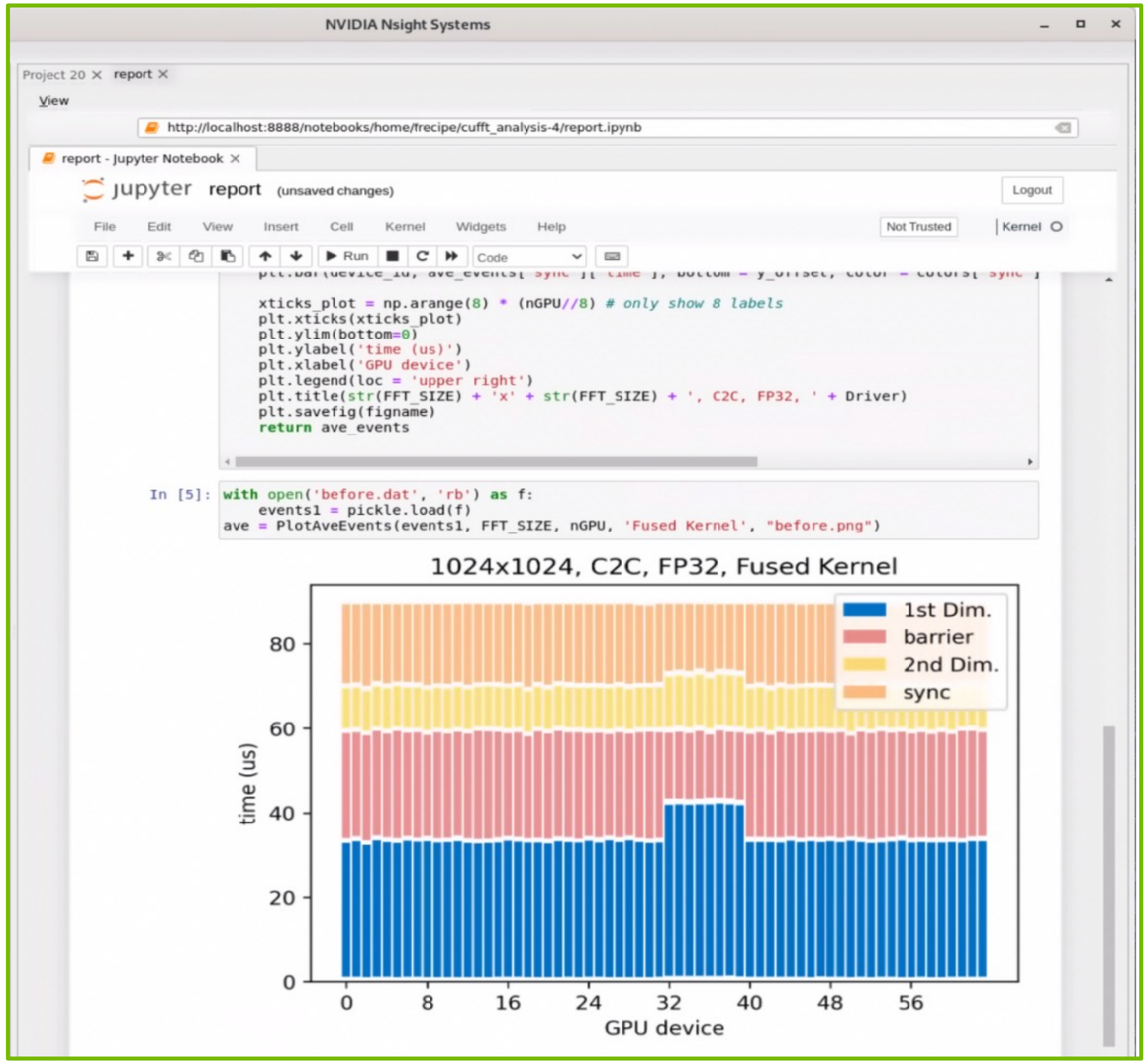


Nsight Systems Multi-Node Analysis

Preview in Nsight Systems 2023.2

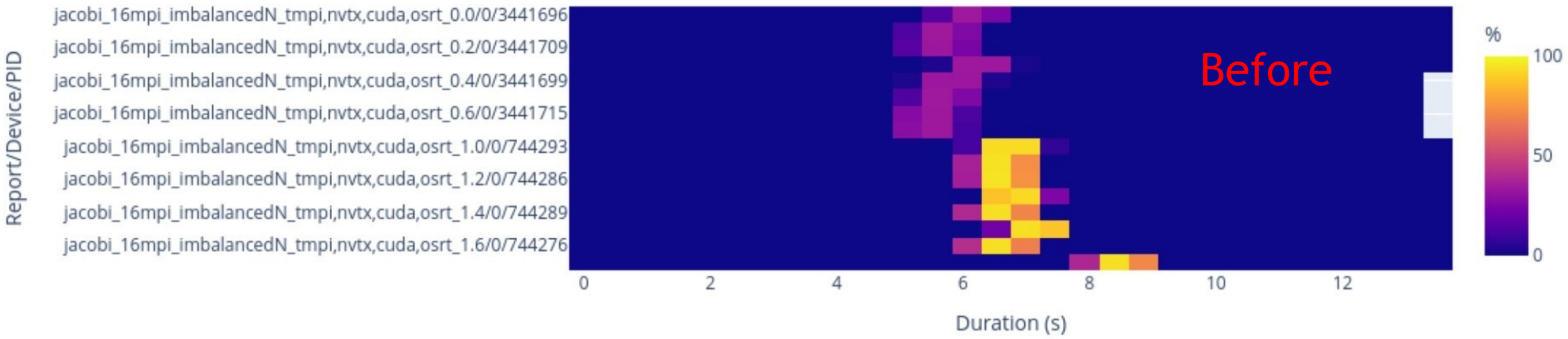
- Recipes to analyze existing reports collected from multiple sources/nodes
- Several default recipes included with Nsight Systems
- Recipes built on extensible Python library
- Recipes generate results various formats:
  - **Jupyter Notebook**
  - Parquet, CSV, …
- Open result in Nsight Systems GUI or Jupyter Lab

```
> ./nsys recipe –help
…
 cuda_api_sum -- CUDA API Summary
 cuda_api_sync -- CUDA Synchronization APIs
 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_memcpy_async -- CUDA Async Memcpy with Pageable Memory
 cuda_memcpy_sync -- CUDA Synchronous Memcpy
 cuda_memset_sync -- CUDA Synchronous Memset
 dx12_mem_ops -- DX12 Memory Operations
 gpu_gaps -- GPU Gaps
 gpu_time_util -- GPU Time Utilization
 nvtx_gpu_proj_trace -- NVTX GPU Trace
 nvtx_sum -- NVTX Range Summary
 osrt_sum -- OS Runtime Summary
…
> ./nsys recipe cuda_gpu_kern_sum --dir ~/profiles
> Generated cuda_gpu_kern_sum-1
```

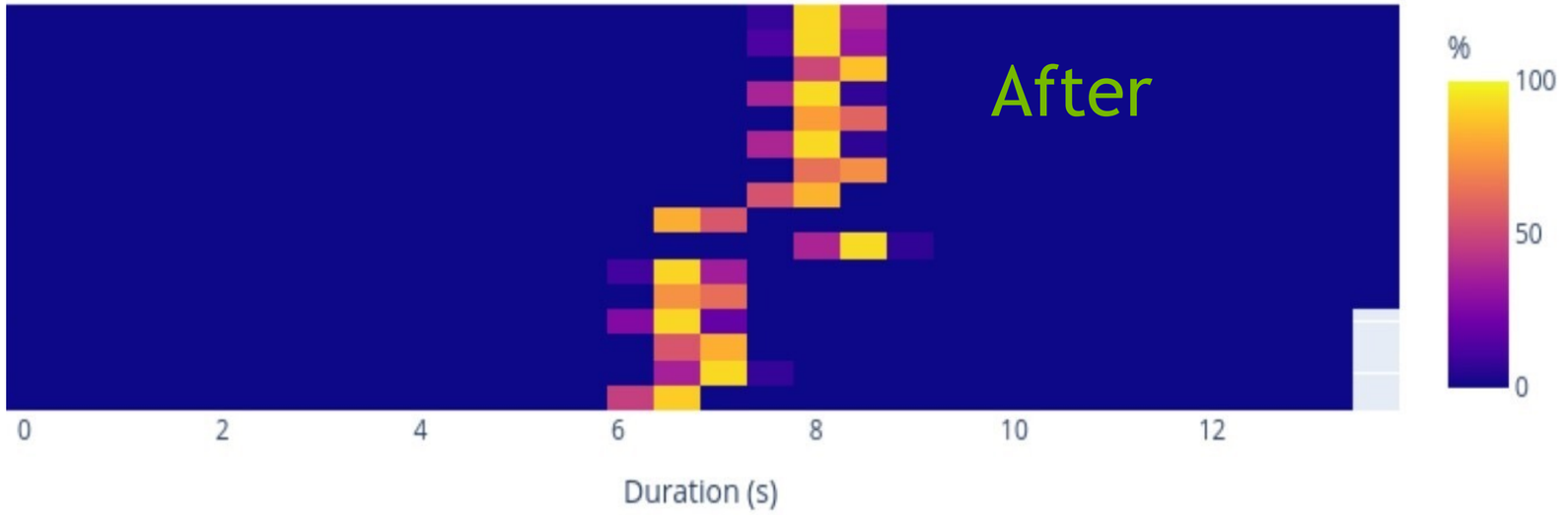


Nsight Systems Multi-Node Analysis

Preview in Nsight Systems 2023.2



- cuda_gpu_time_util_map recipe
- Identify misbehaving GPUs, ranks, or nodes
- Time correlation with application phases
- Click to drill into specific result file for more details



NVTX
Tools Extension API

NVIDIA Tools eXtension

NVTX v3

- 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:

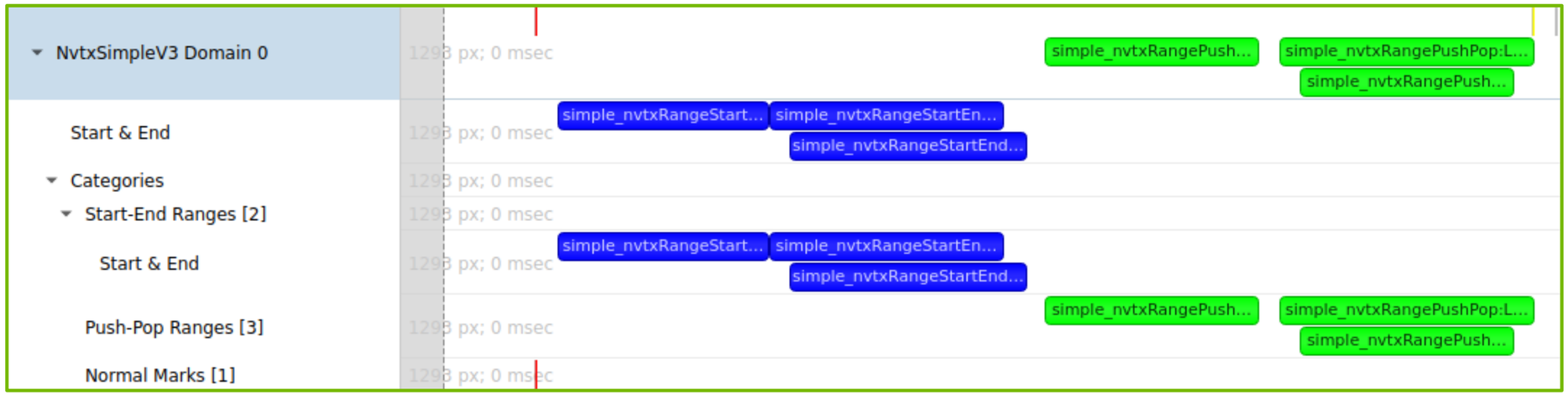
```
nvtxMark("This is a marker");
```

- 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

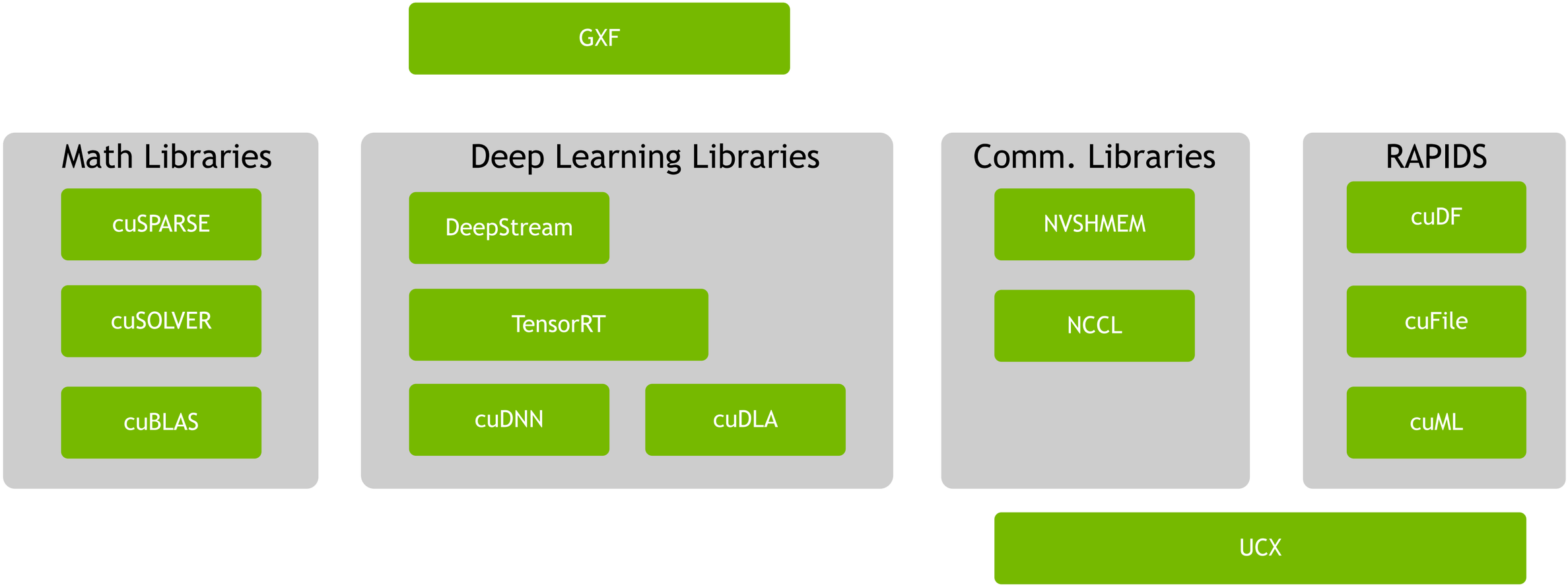
```
nvtxRangeHandle_t handle = nvtxRangeStart("This is a start/end range");
// Somewhere else in the code, not necessarily same thread as Start call:
nvtxRangeEnd(handle);
```



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

NVIDIA SDKs and NVTX

Advanced profiling and performance visualization

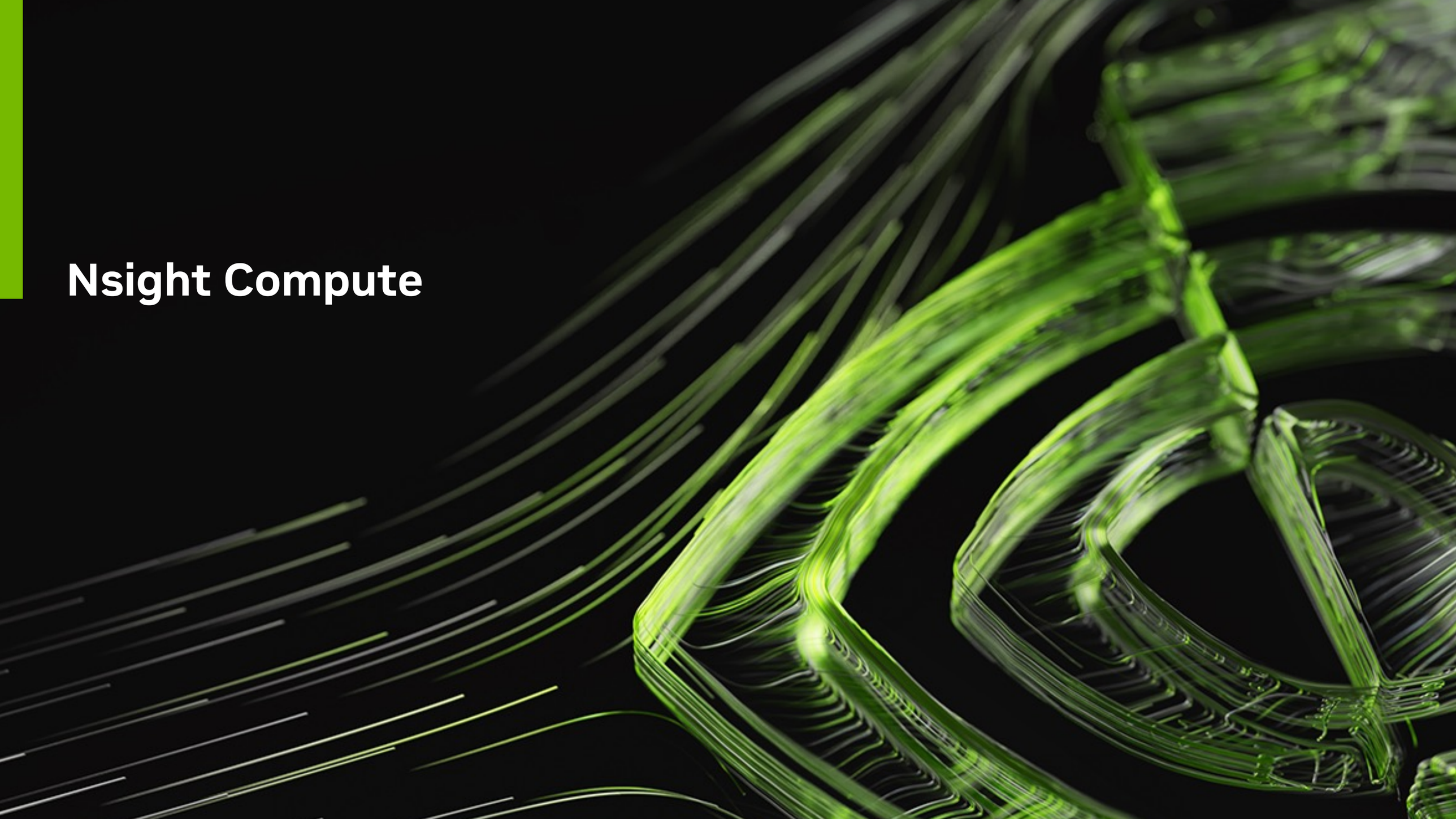


Profiling DL Models

- 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

Nsight Compute



Nsight Compute

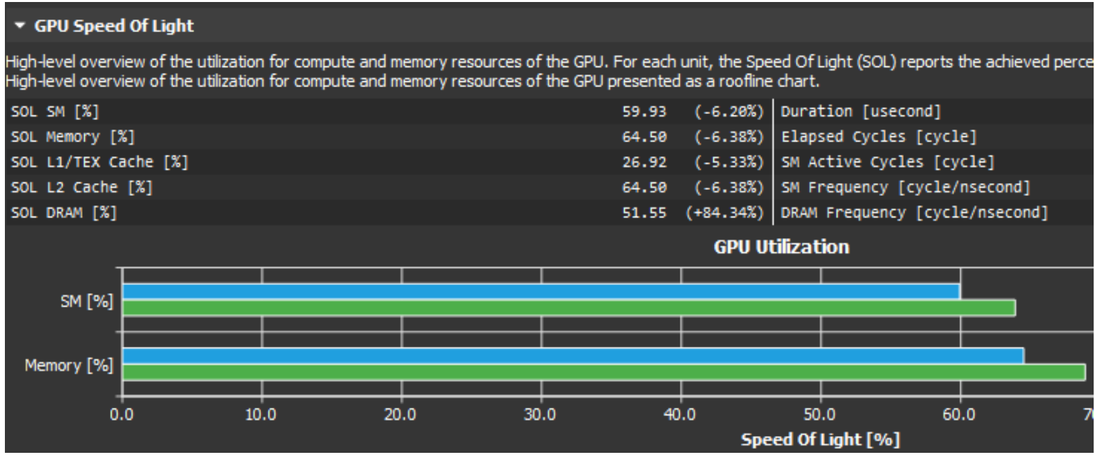
Kernel Profiler

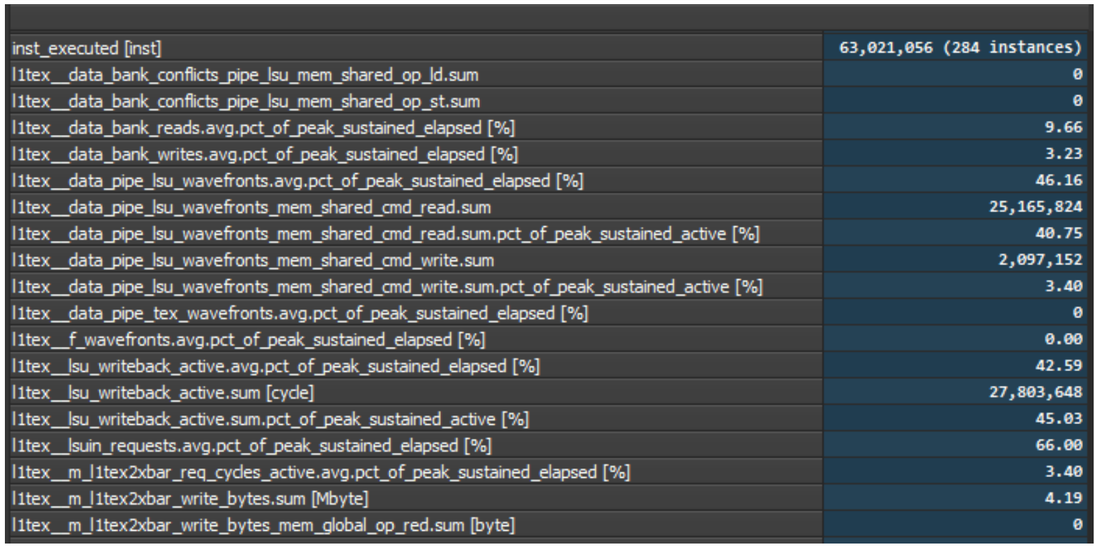
Key Features:

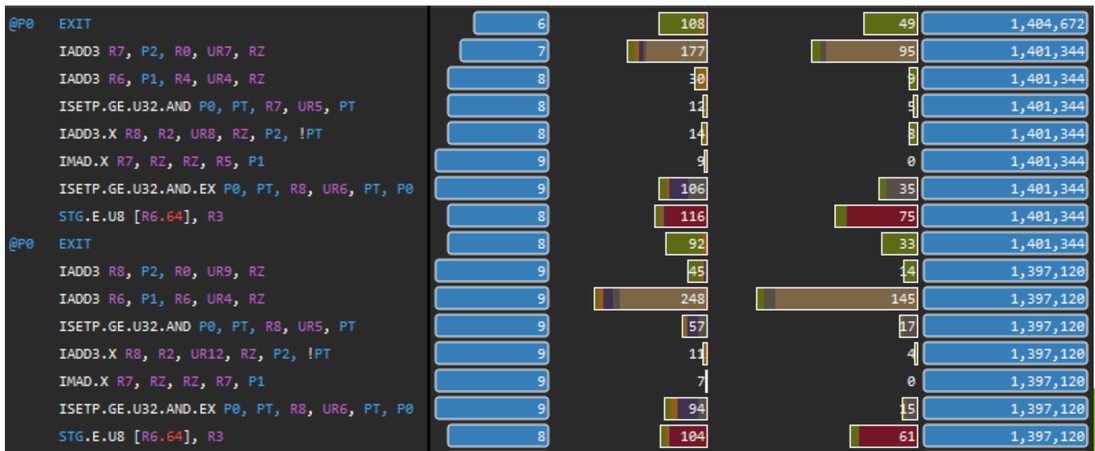
- Interactive CUDA API debugging and kernel profiling
- Built-in rules expertise
- Fully customizable data collection and display
- Command Line, Standalone, IDE Integration, Remote Targets

- OS: Linux (x86, Power, Tegra, Arm SBSA), Windows, macOS X (host only)
- GPUs: Volta+

- Docs/product: https://developer.nvidia.com/nsight-compute





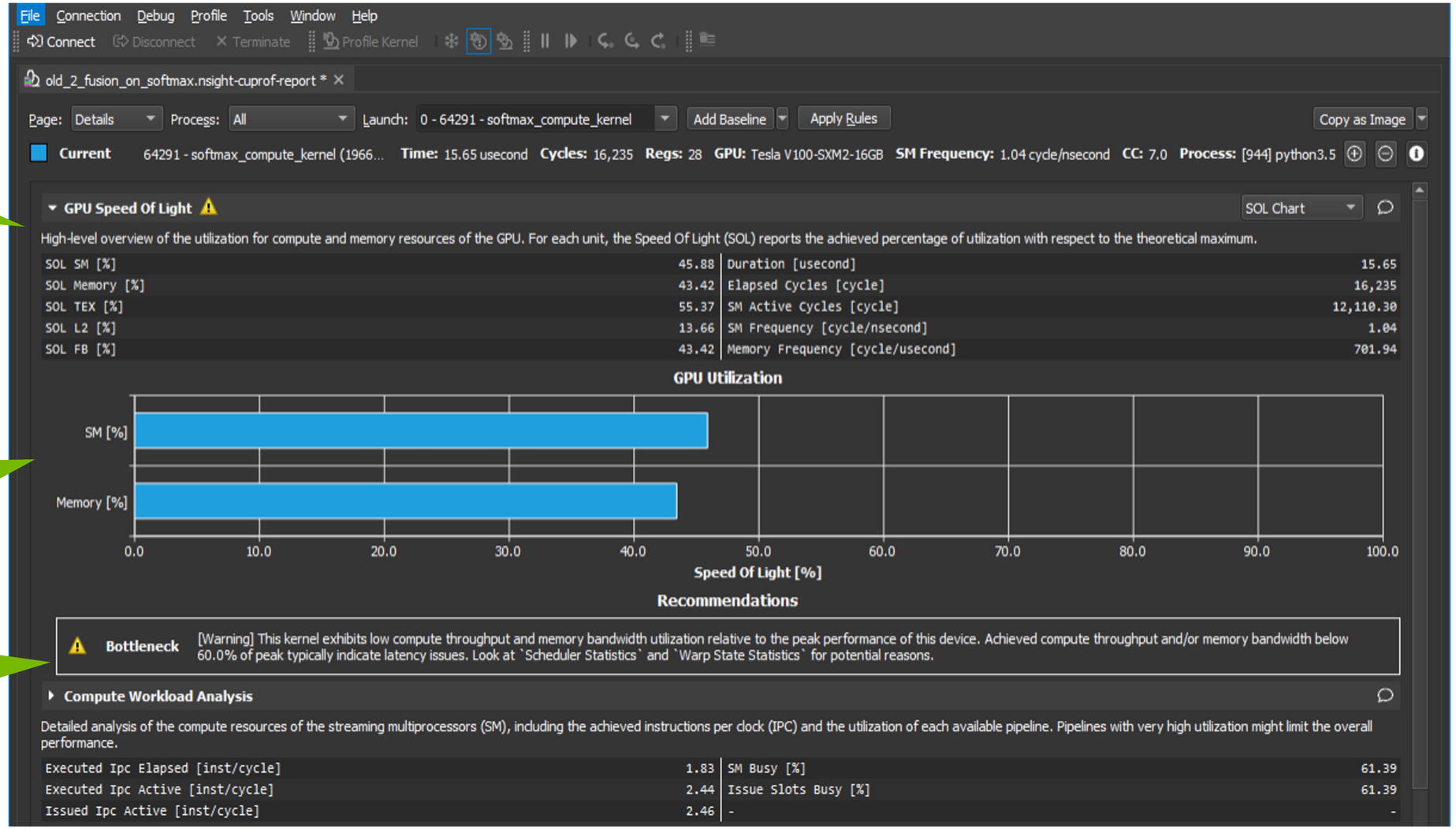


Nsight Compute GUI Interface

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

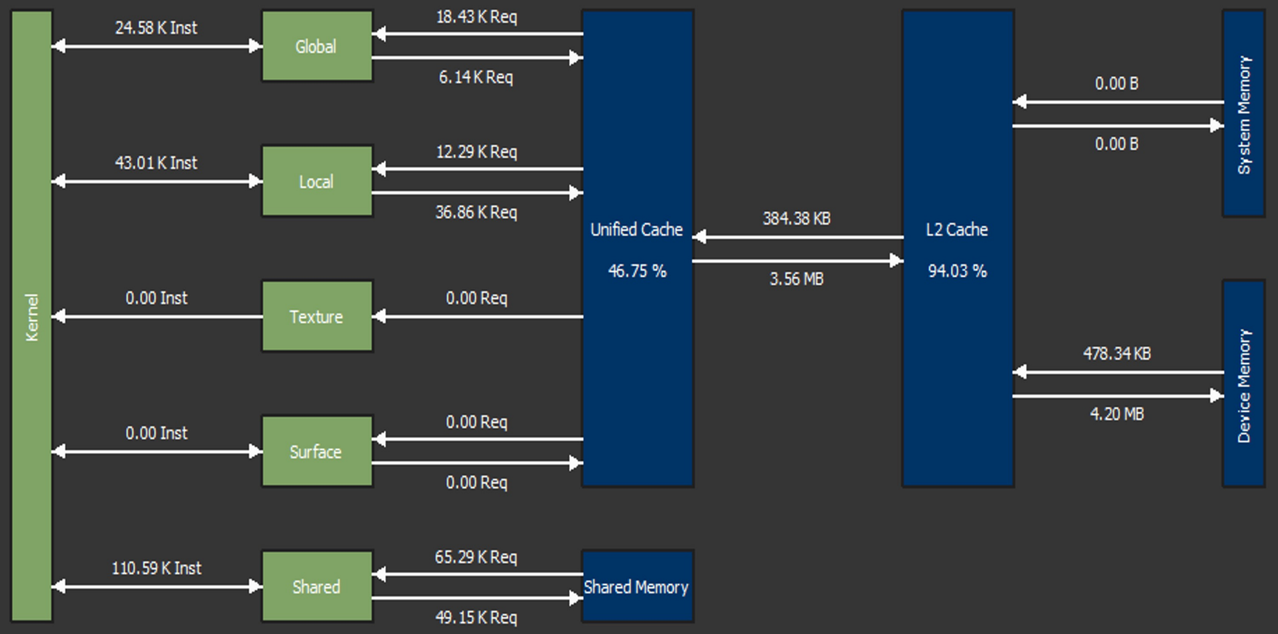
Memory Workload Analysis

All

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.

Memory Throughput [Gbyte/second]	310.08	Mem Busy [%]	42.60
L1 Hit Rate [%]	46.75	Max Bandwidth [%]	44.73
L2 Hit Rate [%]	94.03	Mem Pipes Busy [%]	42.23

Memory Chart

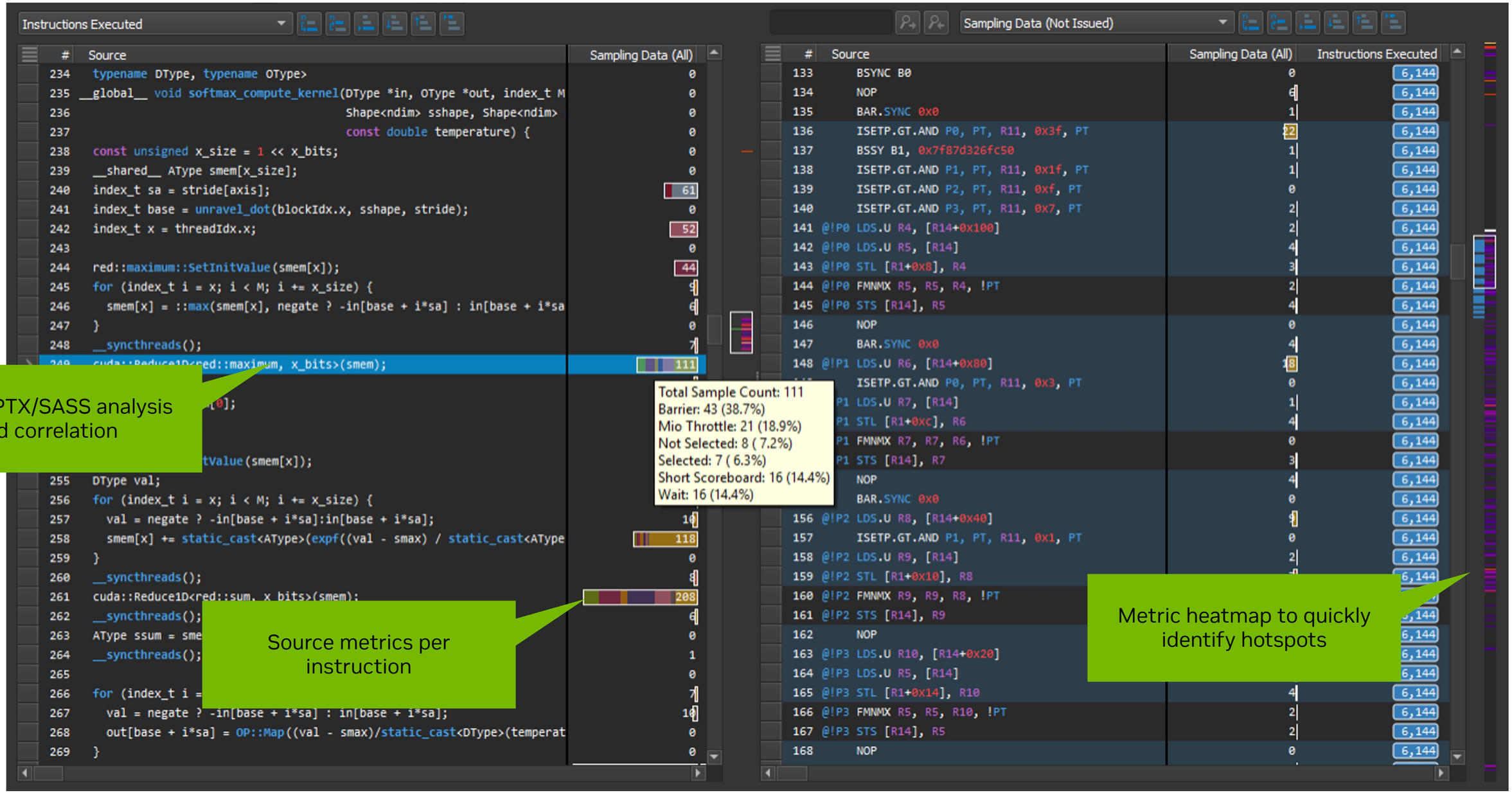


Shared Memory

	Instructions	Requests	% Peak	Bank Conflicts
Shared Load	61,440	65,289	6.59	3,698
Shared Store	49,152	49,152	4.96	0
Shared Atomic	0	-	-	-
Total	110,592	114,441	11.55	3,698

First-Level (Unified) Cache

	Instructions	SM->TEX Requests	% Peak	Hit Rate	TEX->L2 Requests	% Peak	L2->TEX Returns	% Peak
Global Load Cached	18,432	18,432	1.86	66.65	-	-		
Global Load Uncached	-	-	-	-	-	-		
Local Load Cached	12,288	12,288	1.24	100	-	-	12,300	1.24

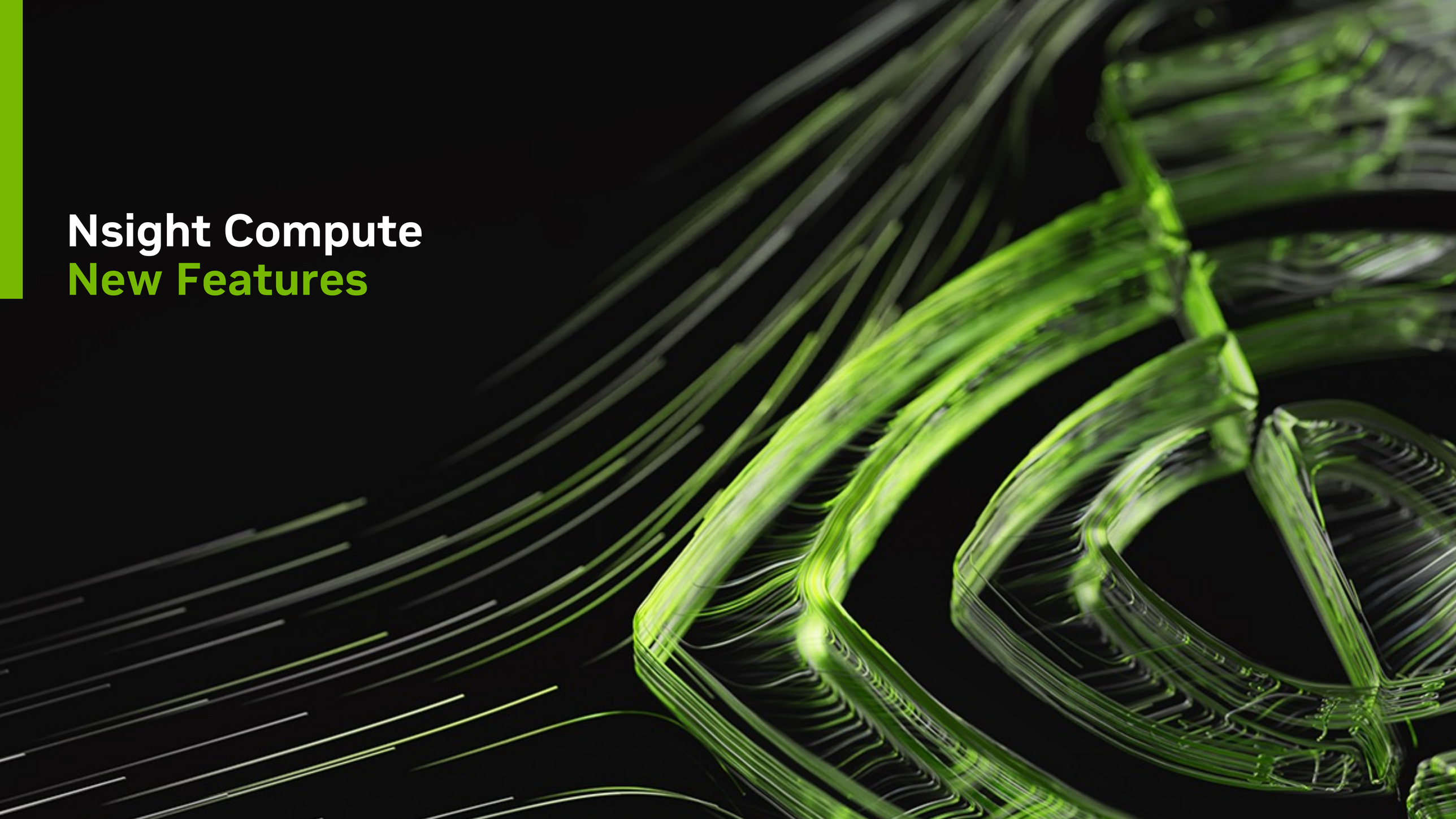

Instructions Executed
Sampling Data (Not Issued)
Source
Sampling Data (All)
Instructions Executed
Total Sample Count: 111
Barrier: 43 (38.7%)
Mio Throttle: 21 (18.9%)
Not Selected: 8 (7.2%)
Selected: 7 (6.3%)
Short Scoreboard: 16 (14.4%)
Wait: 16 (14.4%)
Source/PTX/SASS analysis and correlation
Source metrics per instruction
Metric heatmap to quickly identify hotspots

Kernel profiles with nsight compute

```
$ ncu –k mykernel –o report ./myapp.exe
```

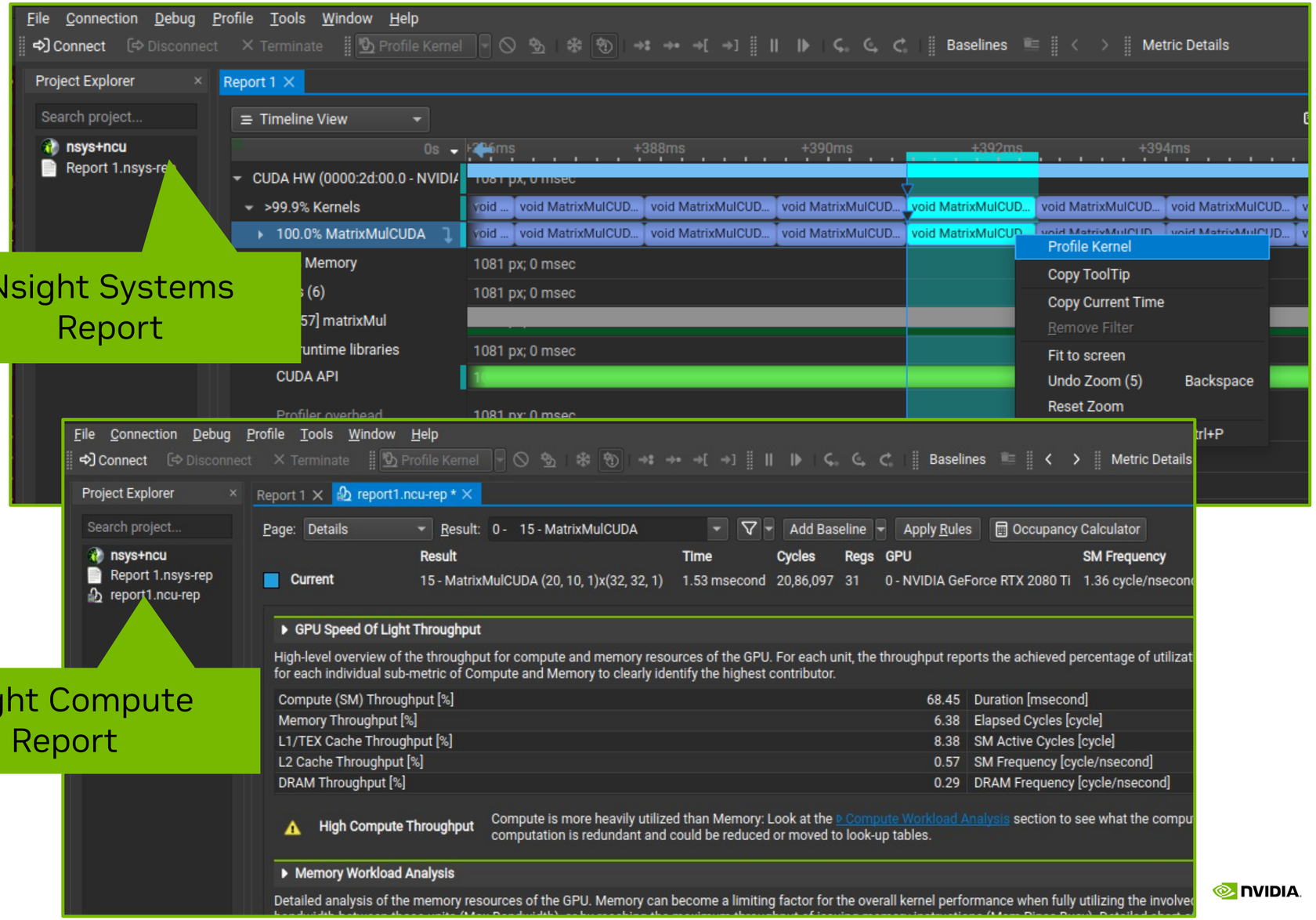
- Generated file: report.ncu-rep
  - Open for viewing in the Nsight Compute UI
- (Without the –k option, Nsight Compute with profile everything and take a long time)

Nsight Compute
New Features



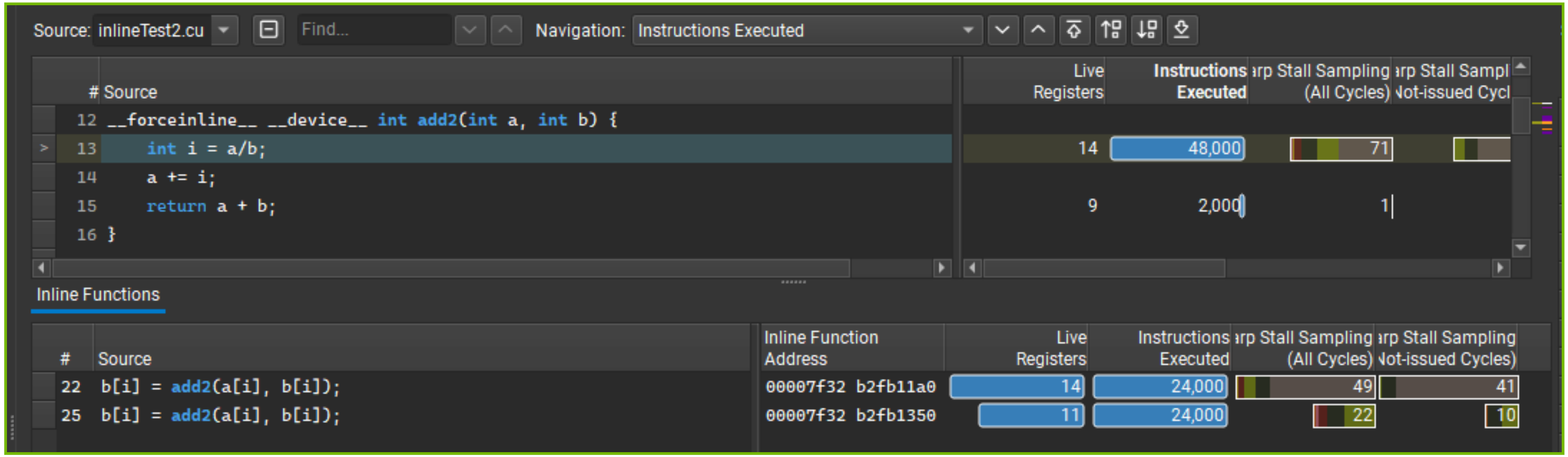
Integrated Basic Trace from Nsight Systems

- 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



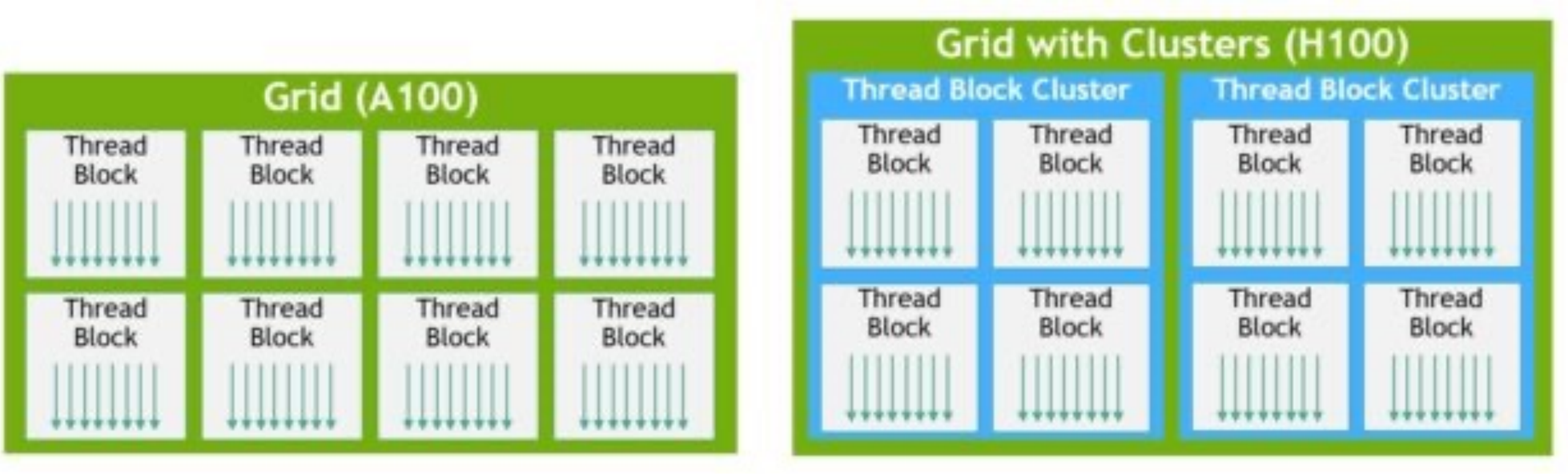
Source Table for Inline Functions

- Metrics can be analyzed per inline site or aggregated for the entire function
- Use compiler *--lineinfo* flag to generate symbols

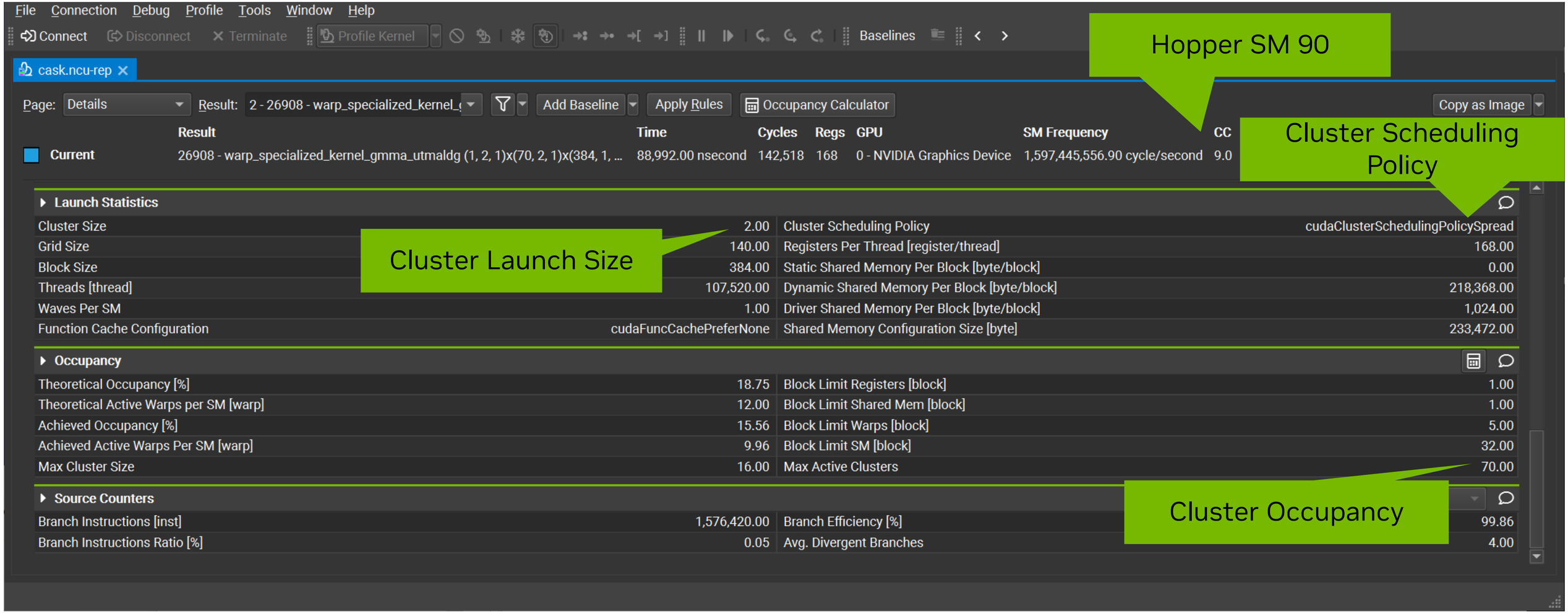


Hopper Thread Block Clusters

- Enables programmatic control of locality at a granularity larger than a single thread block on a single SM
- Clusters enable multiple thread blocks running concurrently across multiple SMs to synchronize and collaboratively fetch and exchange data.
- For more information on Thread Block Clusters see:
  - CUDA: New Features and Beyond
  - CUDA Programming Model for Hopper Architecture



Hopper Thread Block Clusters in Nsight Compute

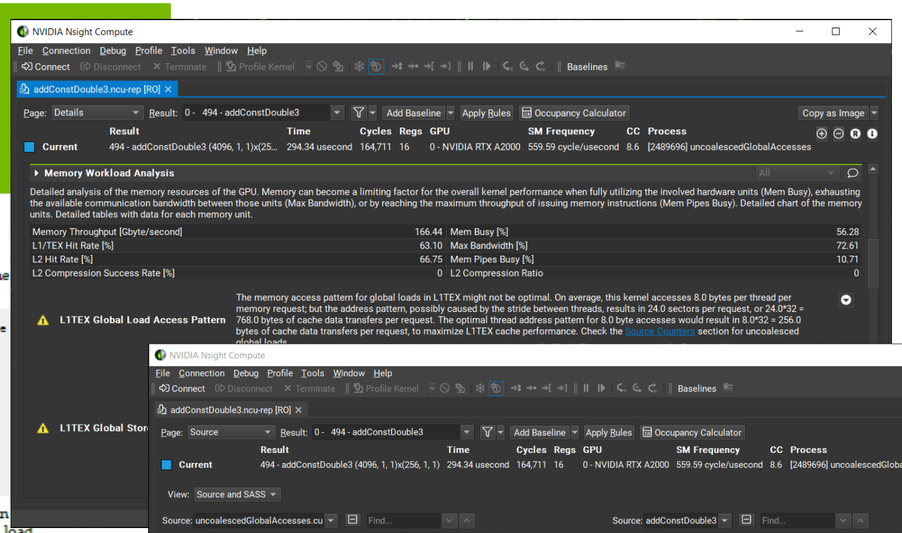


New Documented Samples

Working to Ease the Complexity of Performance Optimization

- Packaged with Nsight Compute installation
- Includes reproducible instructions, descriptions, pre-collected results, and source code
- Explaining common performance issues in CUDA applications
- **Uncoalesced Global Accesses** and **Shared Memory Bank Conflicts**

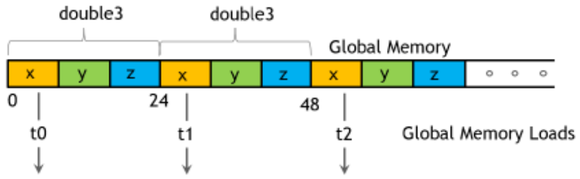
Chapter 4.
INITIAL VERSION OF THE KERNEL



The initial version of the sample code provides a naive implementation for the which adds a floating point constant to an input array of double3.

```
__global__ void addConstDouble3(int numElements, double3 *d_in, double
 double3 *d_out)
{
    int index = blockIdx.x * blockDim.x + threadIdx.x;
    if (index < numElements)
    {
        double3 a = d_in[index];
        a.x += k;
        a.y += k;
        a.z += k;
        d_out[index] = a;
    }
}
```

The instruction `a = d_in[index]` in the kernel code results in each thread in accessing global memory 24-bytes apart. In the first step all threads request a load for `d_in[index].x` as shown in the following diagram. In the second step a load for `d_in[index].y` and in the third step a load for `d_in[index].z` is made by all threads.



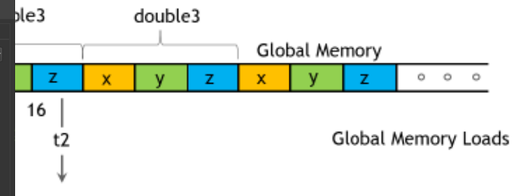
Profile the initial version of the kernel

- Refer to the README distributed with the sample on how to build the application
- Run ncu-ui on the host system

Chapter 5.
UPDATED VERSION OF THE KERNEL

Considering the uncoalesced accesses reported by the profiler we analyze the global load access pattern. Each thread executes 3 reads for the three double values in double3.

We can treat the double3 array as a double array and each thread can process one double double3. With this change threads in a warp access consecutive double loads and stores are coalesced.



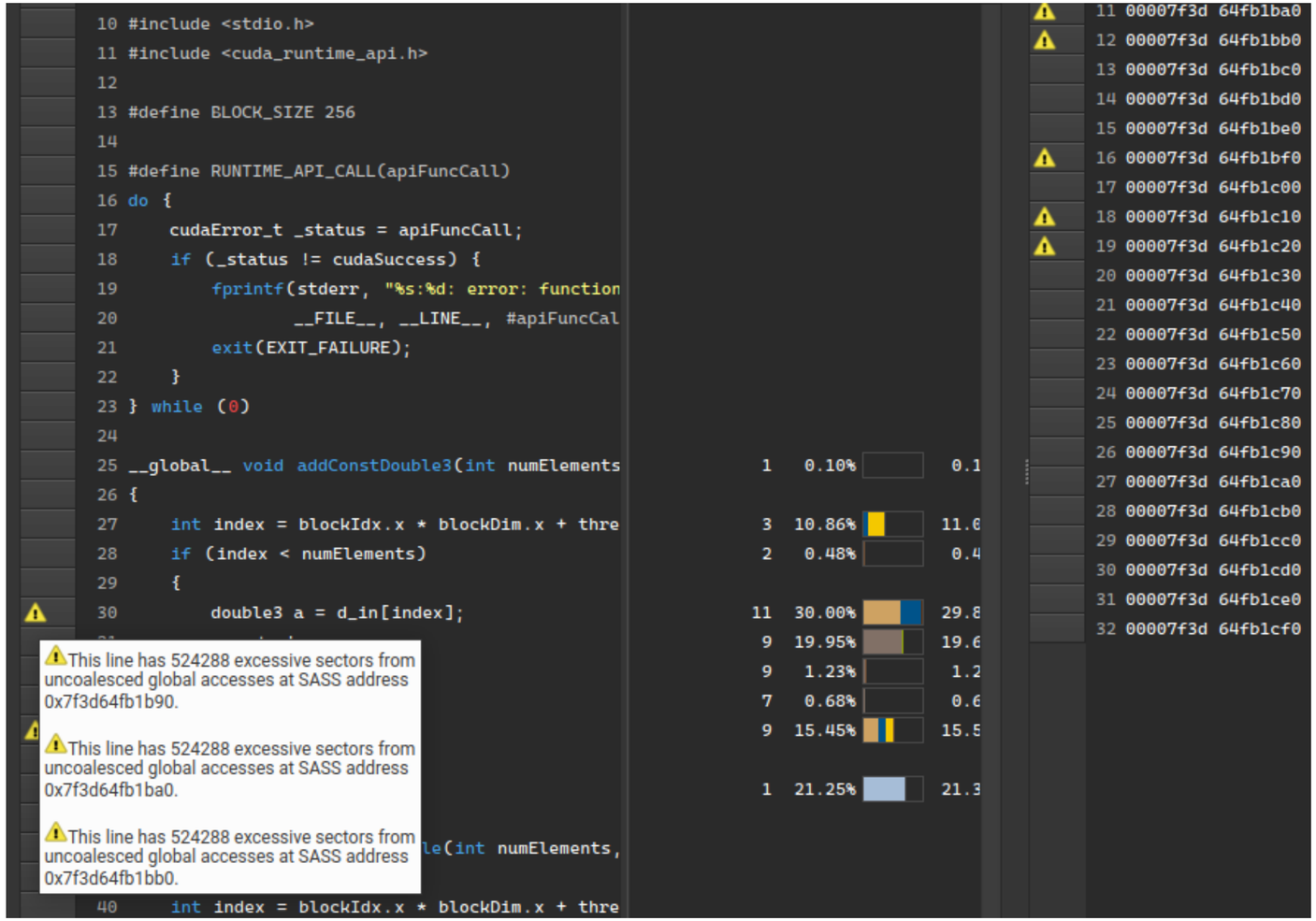

```
d addConstDouble(int numElements, double *d_in, double k, double

= blockIdx.x * blockDim.x + threadIdx.x;
< numElements)

index] = d_in[index] + k;
```

dated kernel

tion has reduced from 294 microseconds to 239 microseconds. We can set initial version of the kernel and compare the profiling results.

Source Page Guidance

- **Branch Navigation**
  - Quickly jump to target of branch instructions
- **Rules Markers**
  - Automated analysis and rules output per source line
  - Quick detection and navigation to lines that matter
  - Explanation of detected issues



Potential Speedup Calculations

- Estimated speedup potential calculated for detected issues
- Sort kernels based on potential speedup
- Helps users understand where to focus effort and potential benefit

ID	Potential Speedup	Issues	Function Name	Demangled Nam	Duration	Compute Throughpu	Memory Throughput	# Registers	Device Name	Process	Grid Size	Block Size
219	18.42	7	bitonicMerge...	bitonicMerge...	44.29	11.19	88.93	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...
106	16.48	7	bitonicMerge...	bitonicMerge...	43.68	11.36	89.84	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...
109	16.24	7	bitonicMerge...	bitonicMerge...	42.82	11.27	89.30	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...
288	15.79	7	bitonicMerge...	bitonicMerge...	42.66	11.48	90.66	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...
52	15.47	7	bitonicMerge...	bitonicMerge...	42.78	11.28	89.41	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...
77	15.45	7	bitonicMerge...	bitonicMerge...	43.30	11.43	90.43	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...
258	15.41	7	bitonicMerge...	bitonicMerge...	44.38	11.23	88.65	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...
79	15.28	7	bitonicMerge...	bitonicMerge...	43.55	11.21	88.65	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...
207	14.14	7	bitonicMerge...	bitonicMerge...	42.88	11.24	88.36	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...
275	14.04	7	bitonicMerge...	bitonicMerge...	42.85	11.12	88.76	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...
150	14	7	bitonicMerge...	bitonicMerge...	44.96	11.08	88.75	22	NVIDIA GeFor...	[1117161] sorting...	2048, 1...	256, 1...

Discovered performance optimization opportunities for this result. See the Details page for more info.

Issue Slot Utilization
Potential Speedup: 16.48x

Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 16.5 cycles. This might leave hardware resources underutilized and may lead to less optimal performance. Out of the maximum of 12 warps per scheduler, this kernel allocates an average of 8.28 active warps per scheduler, but only an average of 0.09 warps were eligible per cycle. Eligible warps are the subset of active warps that are ready to issue their next instruction. Every cycle with no eligible warp results in no instruction being issued and the issue slot remains unused. To increase the number of eligible warps, reduce the time the active warps are stalled by inspecting the top stall reasons on the ▷ Warp State Statistics and ▷ Source Counters sections.

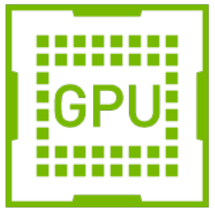
long_scoreboard
Potential Speedup: 5.19x

On average, each warp of this kernel spends 110.1 cycles being stalled waiting for a scoreboard dependency on a L1TEX (local, global, surface, texture) operation. Find the instruction producing the data being waited upon to identify the culprit. To reduce the number of cycles waiting on L1TEX data accesses verify the memory access patterns are optimal for the target architecture, attempt to increase cache hit rates by increasing data locality (coalescing), or by changing the cache configuration. Consider moving frequently used data to shared memory.. This stall type represents about 80.7% of the total average of 136.4 cycles between issuing two instructions.

Putting it all together

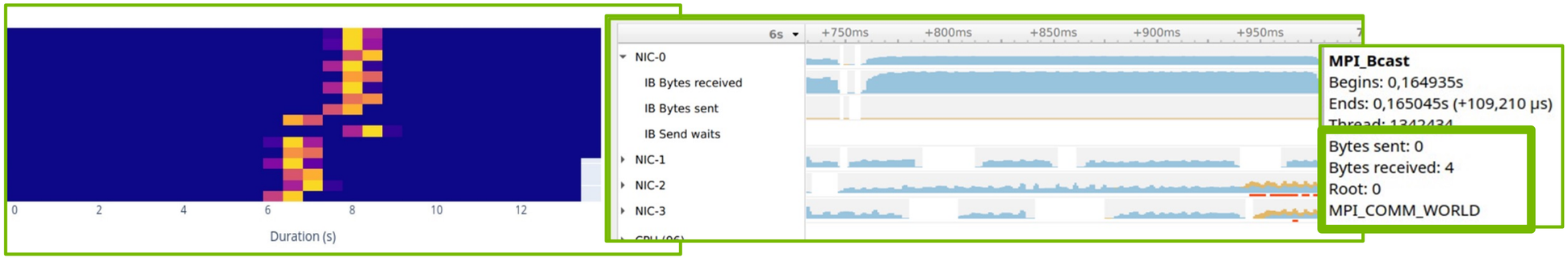
From the Macro to the Micro

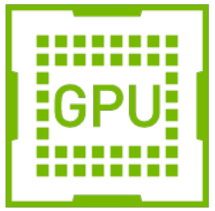
Putting it all together



From the Macro to the Micro

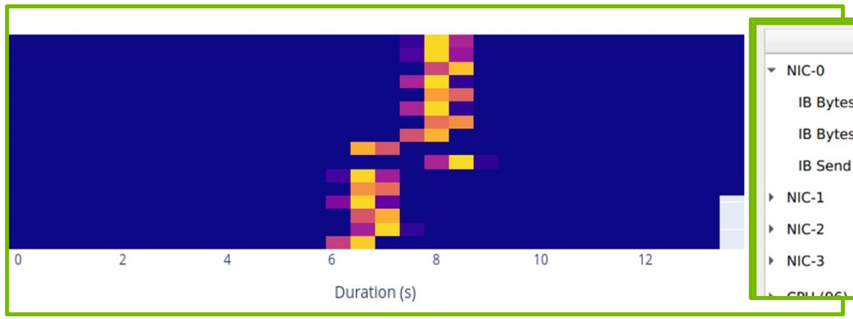
Putting it all together

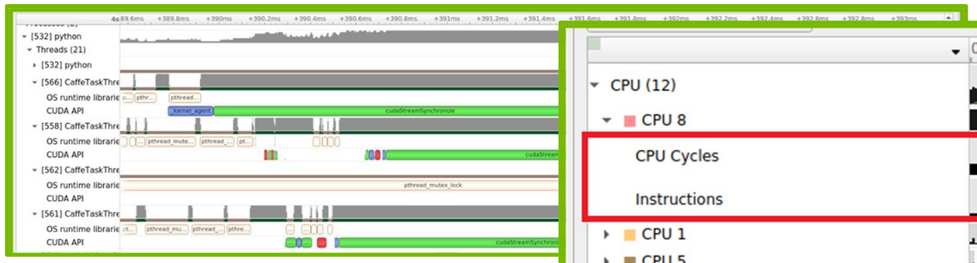


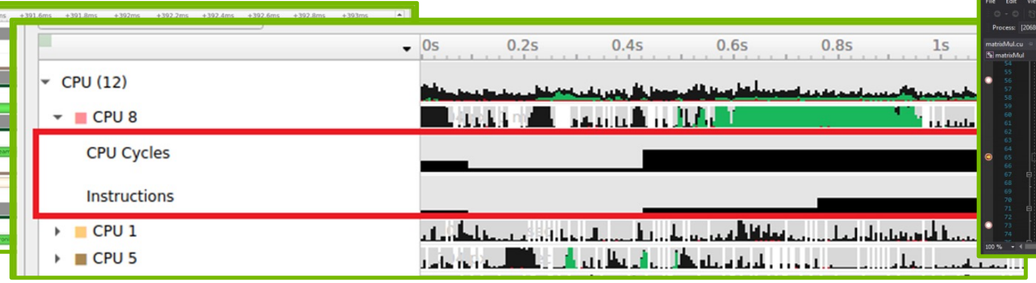


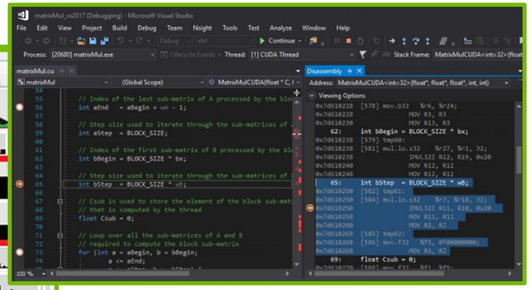
From the Macro to the Micro

Putting it all together



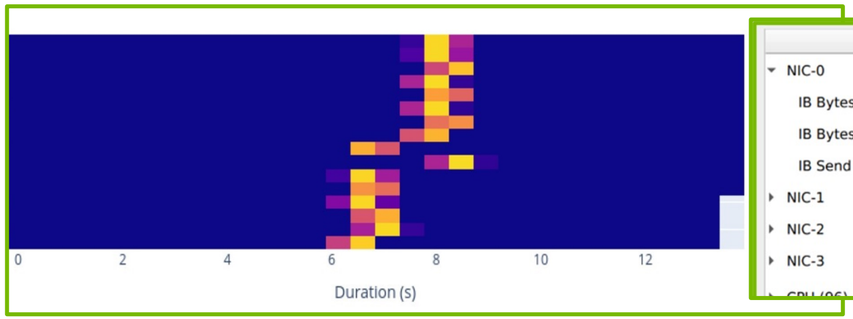


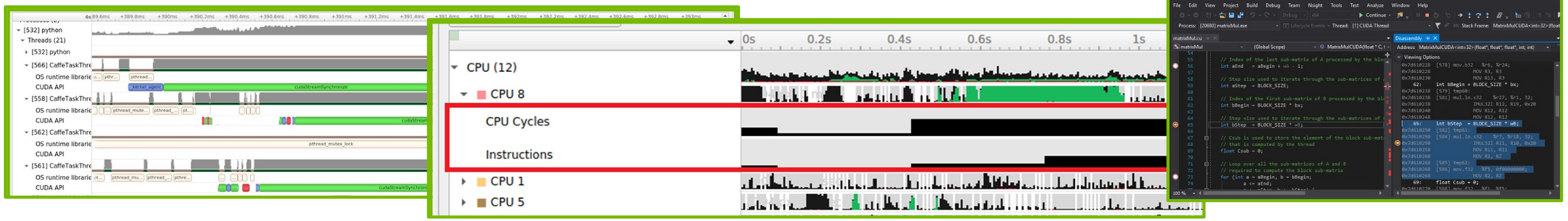


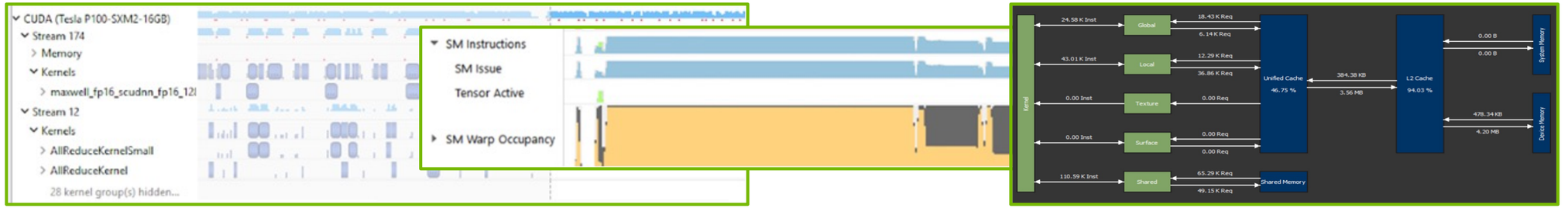


From the Macro to the Micro

Putting it all together

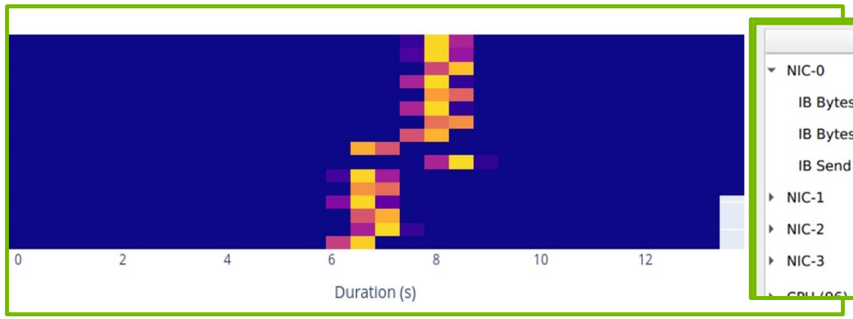


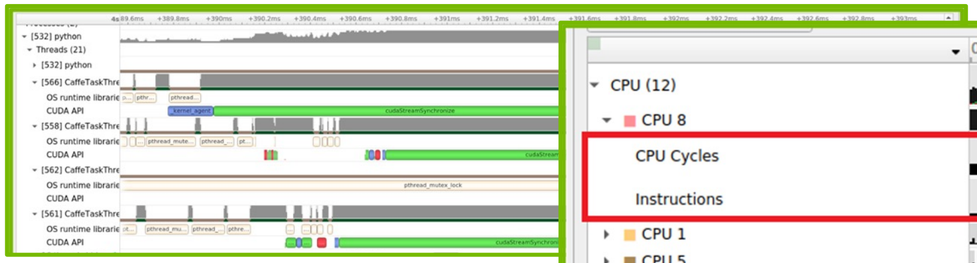


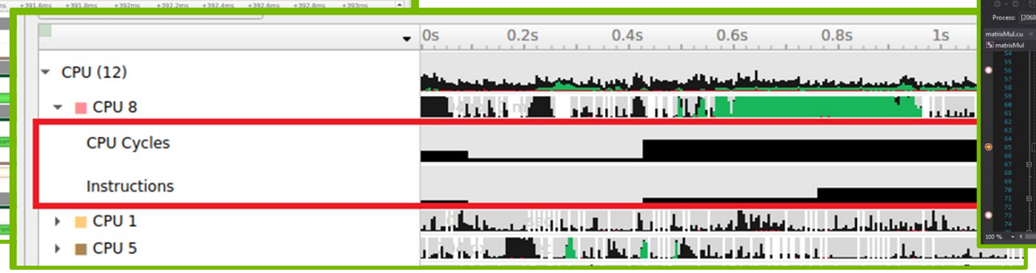


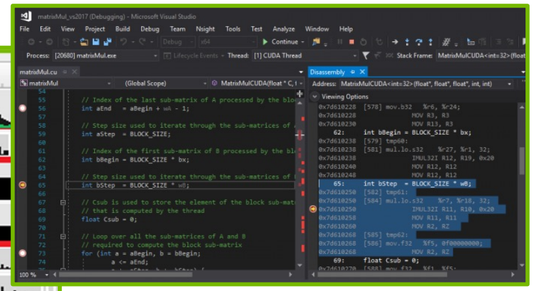
From the Macro to the Micro

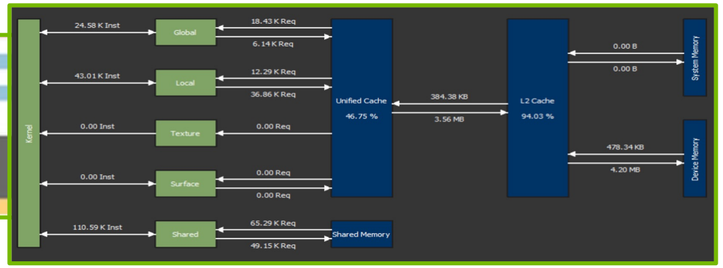
Putting it all together

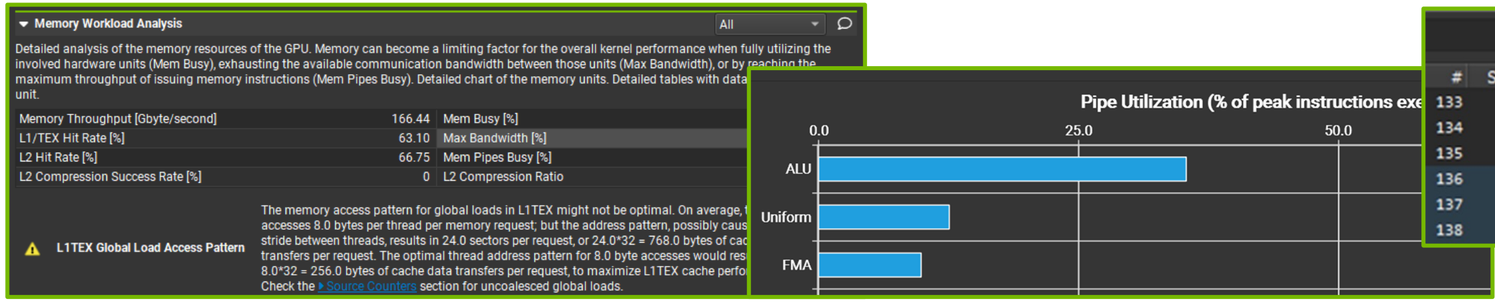


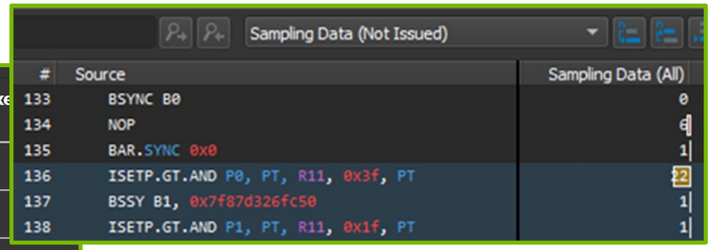












Additional Resources

OpenHackathons.org

OPEN HACKATHONS

Events Attendees Mentors About OpenACC

Search... Log In

Cyfronet AI for Science Bootcamp

Dates: September 11-12, 2023
Event Focus: AI
Event Format: Virtual Event
Region: Europe/Middle East/Africa
Application Status: Open

CESGA AI for Science Bootcamp

Dates: September 18-19, 2023
Event Focus: AI
Event Format: Virtual Event
Region: Europe/Middle East/Africa
Application Status: Open

NASA Open Hackathon

Dates: September 12-21, 2023
Event Focus: HPC+AI
Event Format: Virtual Event
Region: North America/Latin America
Application Status: Closed

CSCS Open Hackathon

Dates: September 18-22, 2023
Event Focus: HPC+AI
Event Format: In-Person Event
Region: Europe/Middle East/Africa
Application Status: Closed

NERSC AI for Scientific Computing Bootcamp

Dates: October 18-20, 2023
Event Focus: AI
Event Format: Virtual Event
Region: North America/Latin America
Application Status: Open

NSM Open Hackathon

Dates: October 10-20, 2023
Event Focus: HPC+AI
Event Format: Hybrid Event
Region: Asia-Pacific
Application Status: Open

Developer Tools Across GTC

- Sessions
  - S51205: From the Macro to the Micro - CUDA Developer Tools Find and Fix Problems at Any Scale
  - S51421: Optimizing at Scale: Investigating and Resolving Hidden Bottlenecks for Multi-Node Workloads
  - S51882: Become Faster in Writing Performant CUDA Kernels using the Source Page in Nsight Compute
  - S51772: Debugging CUDA: An Overview of CUDA Correctness Tools
  - S51230: Orin Performance Bodybuilding with Nsight Developer Tools
  - SE52434: Jetson Edge AI Developer Days: Getting the Most Out of Your Jetson Orin Using NVIDIA Nsight Developer Tools
- Labs
  - DLIT51143: Master Common Optimization Patterns Efficiently with Nsight Profiling Tools
  - DLIT51202: Debugging and Analyzing Correctness of CUDA Applications
  - DLIT51580: Ray-Tracing Development using NVIDIA Nsight Graphics and NVIDIA Nsight Systems
- Connect with Experts
  - CWES52036: What's in Your CUDA Toolbox? CUDA Profiling, Optimization, and Debugging Tools for the Latest Architectures
  - CWES52009: Using NVIDIA Developer Tools to Optimize Ray Tracing
- Developer Tools are free and packaged in the latest version of the CUDA Toolkit
  - https://developer.nvidia.com/cuda-downloads
- Support is available via:
  - https://forums.developer.nvidia.com/c/development-tools/
- More information at:
  - https://developer.nvidia.com/tools-overview


NVIDIA