



Leibniz-Rechenzentrum  
der Bayerischen Akademie der Wissenschaften



DEEP  
LEARNING  
INSTITUTE



DEEP  
LEARNING  
INSTITUTE

# Accelerating CUDA C++ Applications with Multiple GPUs

Dr. Momme Allalen LRZ | 06.02.2025

# Accelerating CUDA C++ Applications with Multiple GPUs



- Learn how to accelerate your **CUDA C++ application** by using multiple GPUs.
- Discover how to combine concurrent computation and memory transfers, computation can be scaled across multiple GPUs without increasing the cost of memory transfers.
- Maximize performance on multi-GPU systems, whether using cloud based servers or on **NVIDIA DGX** systems, these techniques enable you to achieve peak performance from GPU-accelerated applications.
- Implement these single-node, multi-GPU techniques before scaling your applications across multiple nodes.
- Write efficient and correct **CUDA C++** applications. This lecture covers how to write **CUDA C++** applications that efficiently and correctly utilise all available GPUs in a single node, dramatically improving the performance of your applications and making the most cost-effective use of systems with multiple GPUs.

# Workshop Webpage



DEEP  
LEARNING  
INSTITUTE



- **Lecture material will be made available under:**
  - <https://tinyurl.com/hdli1w25>



# THE NSIGHT SUITE COMPONENTS

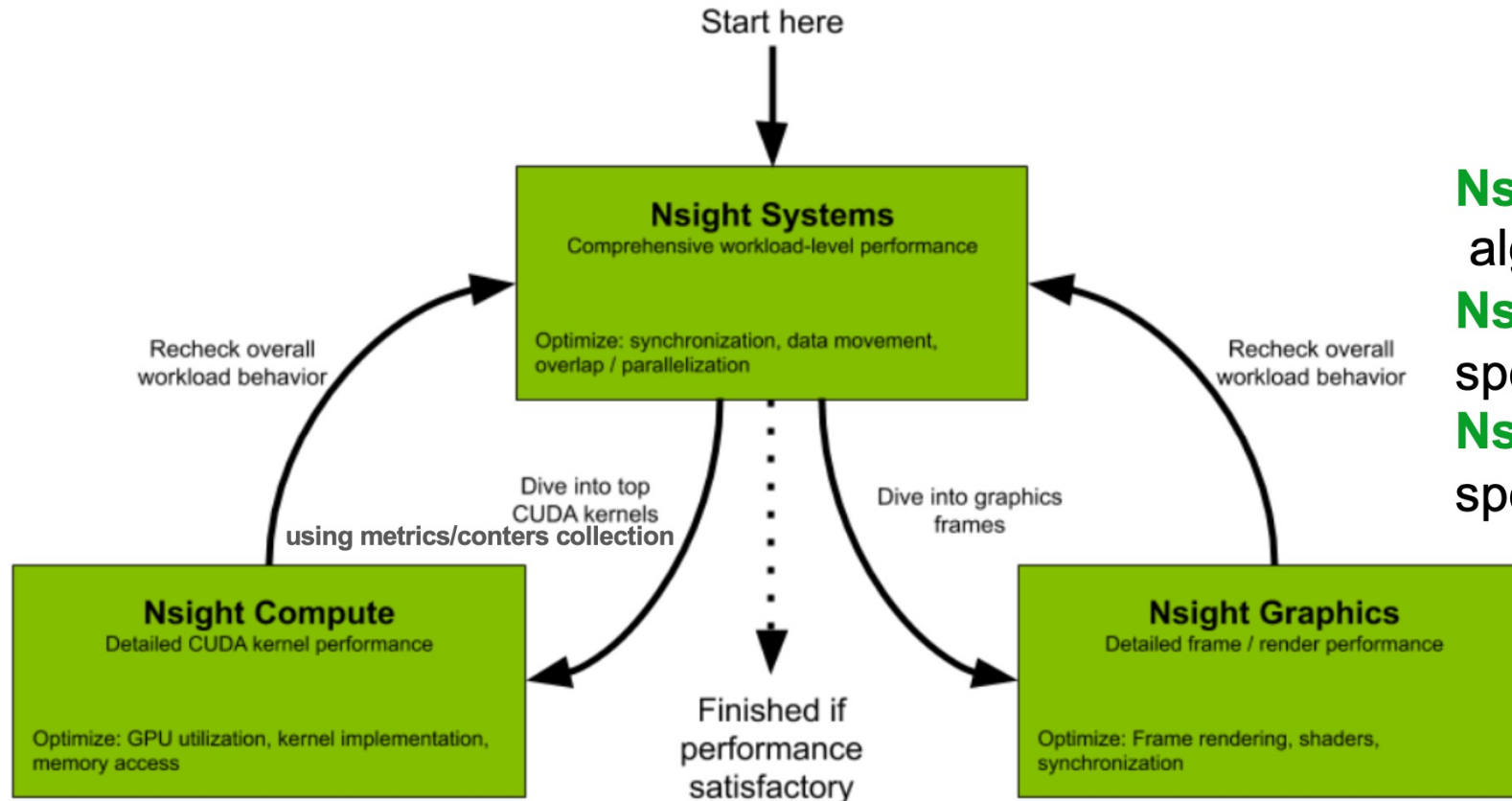


Figure 1. Flowchart describing working with new NVIDIA Nsight tools for performance optimization

**Nsight Systems** – Analyze application algorithm system wide

**Nsight Compute** – Debug/ & optimize specific CUDA kernels

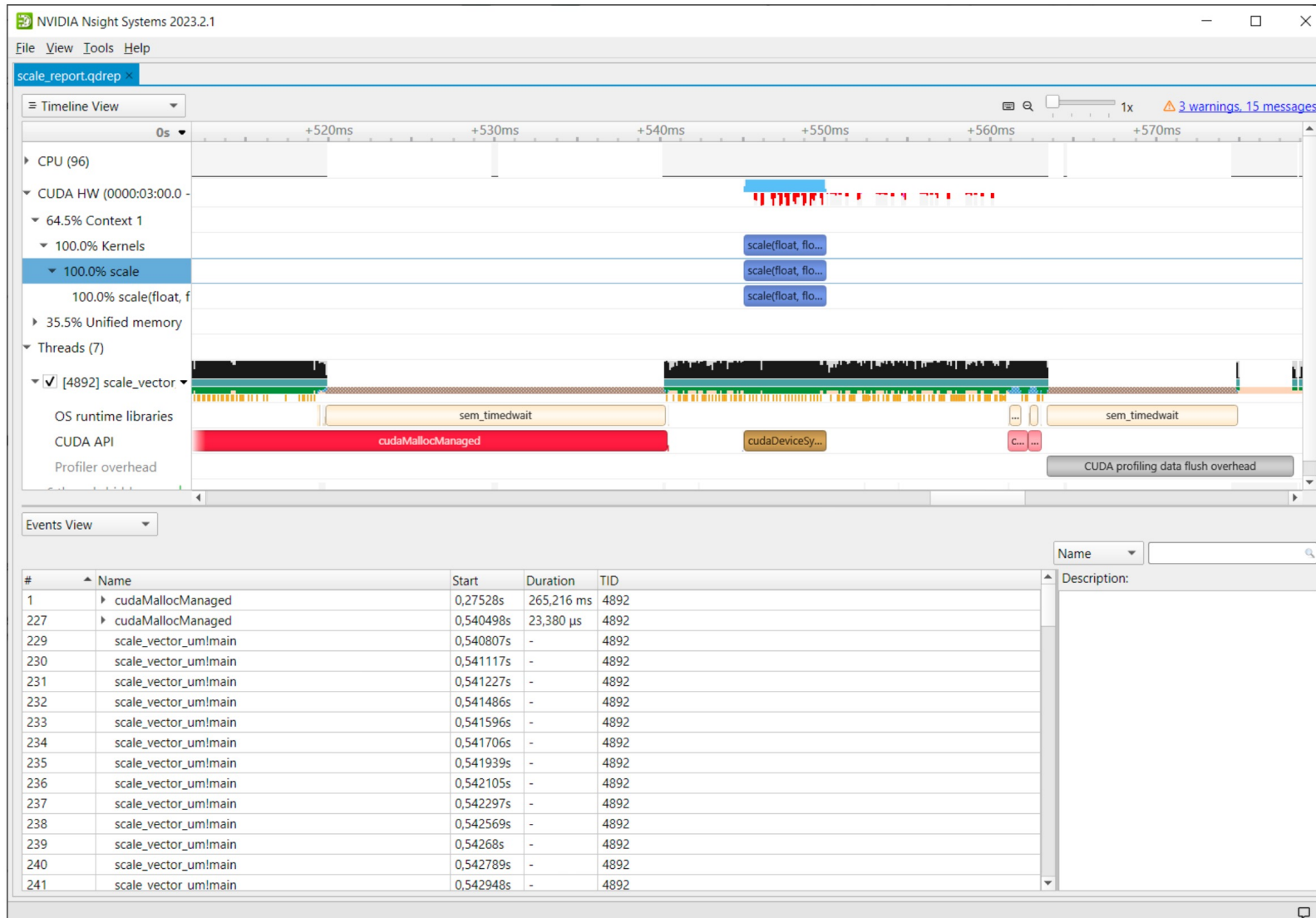
**Nsight Graphics** – Debug/ & optimize specific graphics workloads

**nvprof** replaced with **nsys –profile....**

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

# Nsight Systems GUI

## Main timeline view, Events View



- Nsight Systems is an extremely low overhead profiling Tool across any number of CPUs and GPUs.
- Nsight System is your first stop on your profiling workflow, inspect your algorithm timing and GPU interaction and identify a large number of opportunities for optimization.

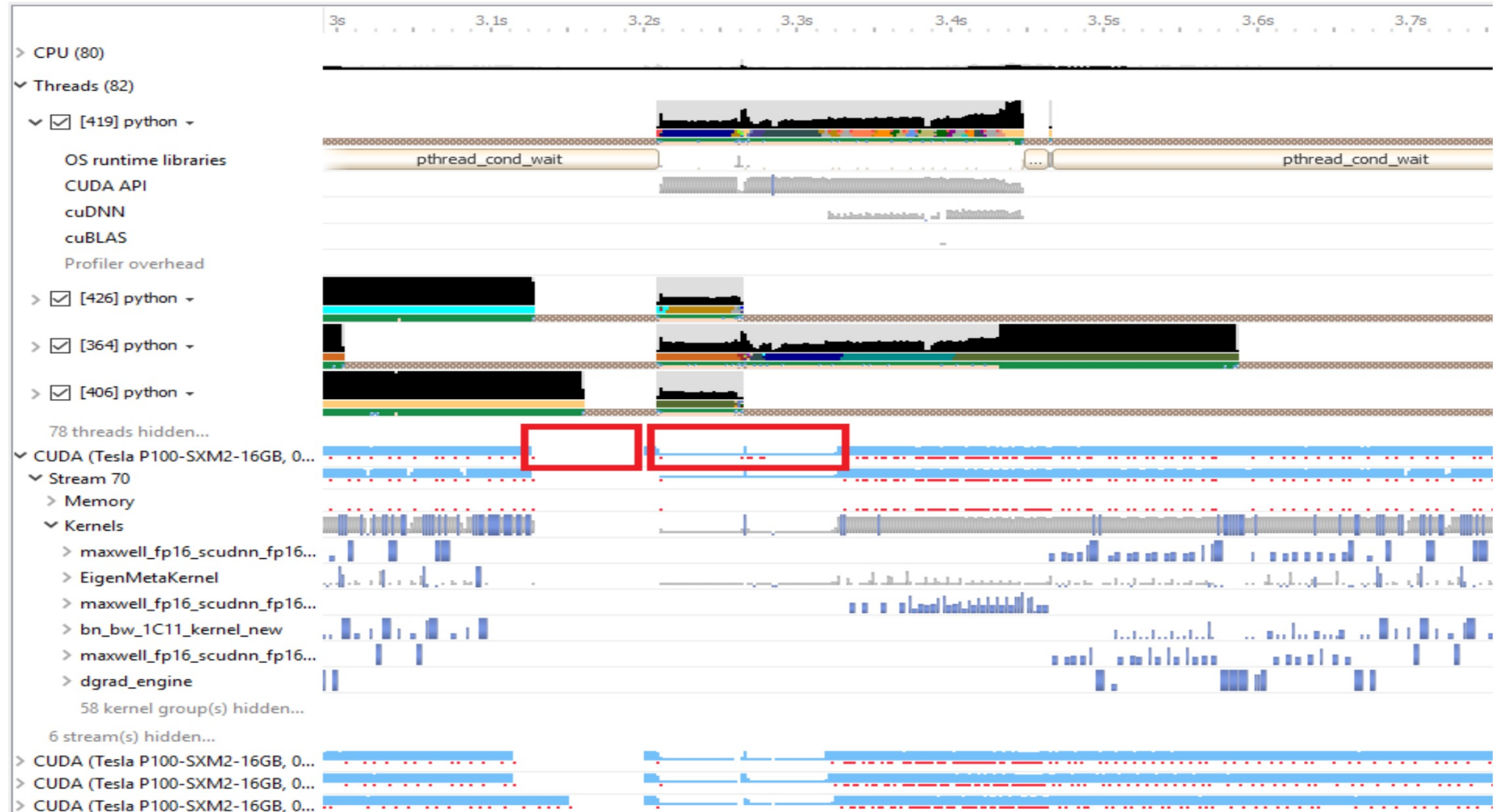
# NSIGHT SYSTEMS



- Provides users with a more complete view of how their codes balance workload across multiple CPUs and GPUs
- Locate optimization opportunities, helps and allows to identify issues such as:
  - *GPU starvation*
  - *Insufficient CPU parallelisation or pipelining*
  - *Unexpectedly expensive CPU or GPU algorithm*
  - *Unnecessary GPU synchronization*
- The tool uses low overhead tracing and sampling techniques to collect process and thread activity and visualize millions of events on a very fast GUI timeline
- Correlates that data across CPU cores and GPU streams, allowing users to investigate bottlenecks.
- Multi-platform: Linux & Windows, x86-64, Tegra, Power, MacOSX (host only)

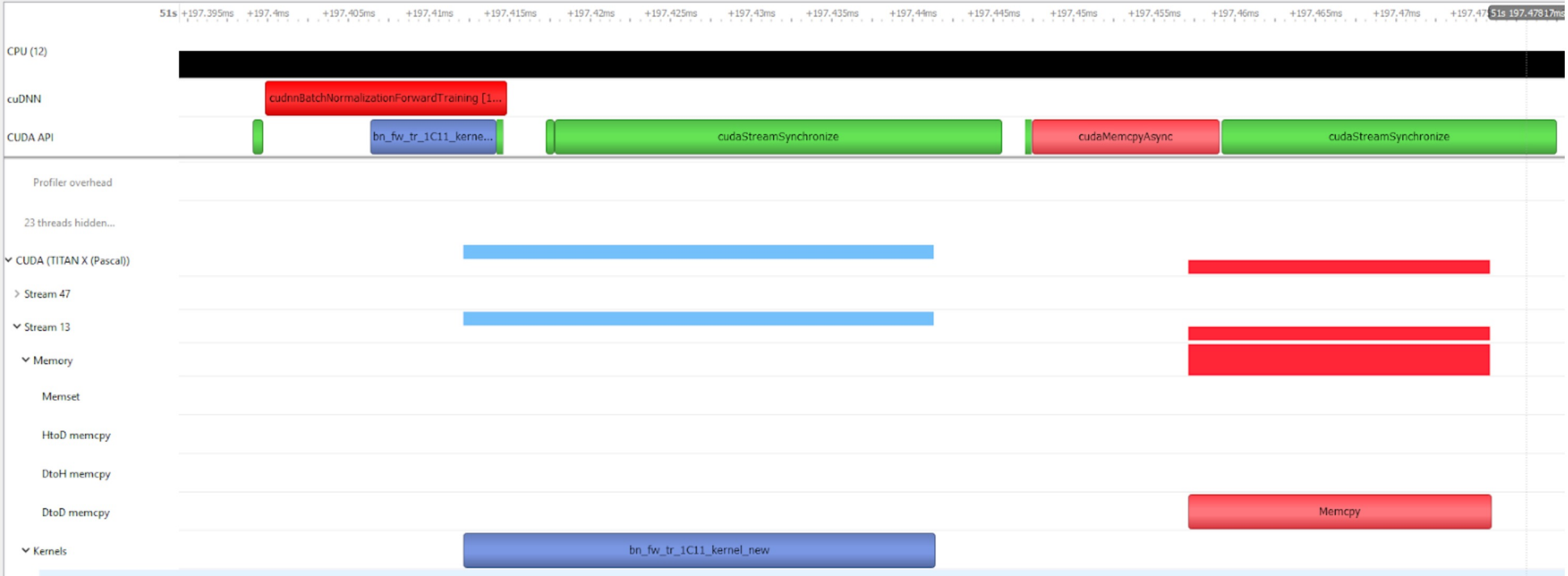
<https://developer.nvidia.com/nsight-systems>

# GPU Starvation Investigations



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

# Unnecessary GPU Synchronisation Calls



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

# NVIDIA NSIGHT SYSTEMS



- Support: **MPI**, **OpenACC**, **OpenMP**
- Complex data mining capabilities, enables to go beyond basic statistics.
- Support multiple simultaneous sessions.
- **MPI trace** feature enables to analyse when the threads are busy or blocked in long-running functions of the **MPI** standard, available on **OpenMPI**, **MPICH** and **NVShmem**.
- **OpenACC** trace enables to see where code has been offload and parallelized onto the GPU, which helps you to analyse the activities executing on the CPUs and GPUs in parallel.
- Tracing **OpenMP** code is available for compilers supporting **OpenMP5** and **OMPT interface**. This capability enables tracing of the parallel regions of code that are distributed either across multiple threads or to the GPU.
- Provides support for **CUDA** graphs. To understand the execution of the source of **CUDA** kernels and execution of **CUDA** graphs, kernels can be correlated back through the graph launch, instantiation, and all the way back to the code creation, to identify the origin of the kernel execution on the GPU.



# NVIDIA NSIGHT COMPUTE

## Important Features



DEEP  
LEARNING  
INSTITUTE



- Result comparison across one or multiple reports within the tool.
- **Graphical profile report** for performance visualization.
- Interactive kernel profiler and API debugger, enabling simultaneous debugging of CPU and GPU while handling thousands of concurrent threads.
- **Fast data collection** for efficient performance analysis.
- **GUI and command-line interface** for flexible user interaction.
- **Fully customizable reports and analysis rules** to tailor insights to specific needs

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

# Command Line Options nsys



Command	Description
profile	A fully formed profiling description requiring and accepting no further input. The command switch options used (see below table) determine when the collection starts, stops, what collectors are used (e.g. API trace, IP sampling, etc.), what processes are monitored, etc.
start	Start a collection in interactive mode. The start command can be executed before or after a launch command.
stop	Stop a collection that was started in interactive mode. When executed, all active collections stop, the CLI process terminates but the application continues running.
cancel	Cancels an existing collection started in interactive mode. All data already collected in the current collection is discarded.
launch	In interactive mode, launches an application in an environment that supports the requested options. The launch command can be executed before or after a start command.
shutdown	Disconnects the CLI process from the launched application and forces the CLI process to exit. If a collection is pending or active, it is cancelled
export	Generates an export file from an existing .nsys-rep file. For more information about the exported formats see the /documentation/nsys-exporter directory in your Nsight Systems installation directory.
stats	Post process existing Nsight Systems result, either in .nsys-rep or SQLite format, to generate statistical information.
analyze	Post process existing Nsight Systems result, either in .nsys-rep or SQLite format, to generate expert systems report.
status	Reports on the status of a CLI-based collection or the suitability of the profiling environment.
sessions	Gives information about all sessions running on the system.

<https://docs.nvidia.com/nsight-systems/UserGuide/index.html>



## Interactive CUDA Kernel profiler

- **Targeted metric sections** for various performance aspects (Debug/Profile).
- **API debugging** via a user interface command-line tool.
- High-frequency GPU performance counters, customizable data collection and presentation (tables, charts ..).
- **Python-based rules** for guided analysis or postprocessing.
- **Customizable, data-driven user interface** for metric collection, extendedable with analysis scripts for post-processing results.

<https://docs.nvidia.com/nsight-compute/NsightCompute/index.html>

# Nsight Compute Feature Spotlight in CUDA Toolkit 11 and A100

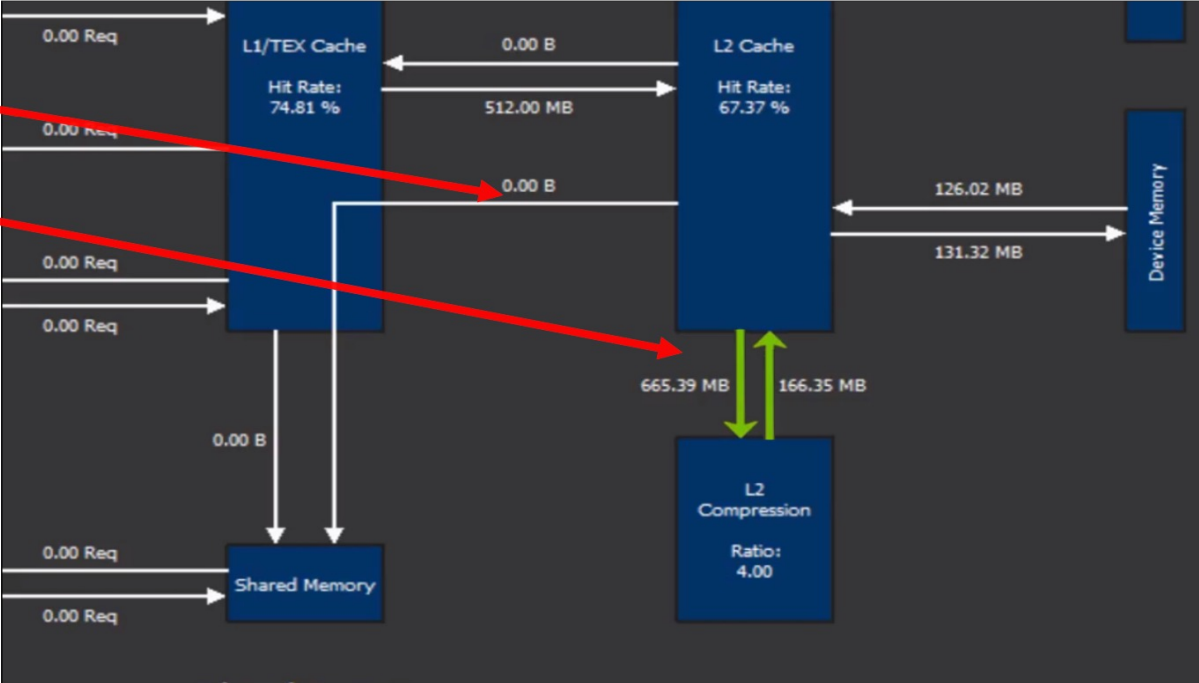
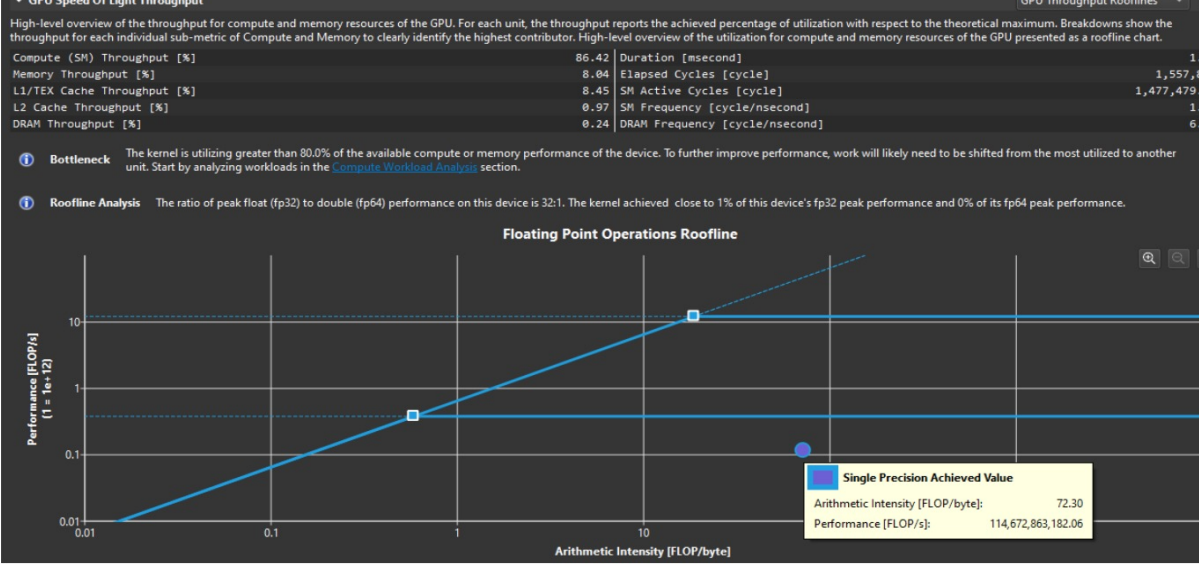
- **Roofline Analysis**

Arithmetic intensity = Compute/Memory  
 FLOPS = Floating Points Ops/Second

- **Asynchronous copy**

- **Sparse Data Compression**

Shows the amount of data compressed through this feature and the compression ratio, helps on kernels with bandwidth or cache issues.



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

# NVIDIA® Tools Extension SDK (NVTX)



- C-based Application Programming Interface (API) for annotating events, code ranges, and resources in your applications
- Codes which integrate NVTX can use NVIDIA Nsight, Tegra System Profiler, and Visual Profiler to capture and visualize these events and ranges.

```
[allalen1@jwlogin22 v2]$ ncu -h | grep nvtx
--nvtx                Enable NVTX support.
--nvtx-include arg    Adds include statement to the NVTX filter, which allows selecting kernels to
--nvtx-exclude arg    Adds exclude statement to the NVTX filter, which allows selecting kernels to
--print-nvtx-rename arg (=none) Select how NVTX should be used for renaming:
                        per-nvtx
Usage of --nvtx-include and --nvtx-exclude:
ncu --nvtx --nvtx-include "Domain A@Range A"
ncu --nvtx --nvtx-exclude "Range A]"
ncu --nvtx --nvtx-include "Range A" --nvtx-exclude "Range B"
```

<https://docs.nvidia.com/nsight-visual-studio-edition/nvtx/index.html>

# NVIDIA® Tools Extension SDK (NVTX)



```
#include <nvToolsExt.h>
#include <sys/syscall.h>
#include <unistd.h>
```

```
static void wait(int seconds) {
    nvtxRangePush(__FUNCTION__);
    nvtxMark("Waiting...");
    sleep(seconds);
    nvtxRangePop();
}
```

```
int main(void) {
    nvtxNameOsThread(syscall(SYS_gettid), "Main Thread");
    nvtxRangePush(__FUNCTION__);
    wait(1);
    nvtxRangePop();
}
```

**nsys profile -t nvtx --stats=true ...**

**Or for Julia code:**

**nsys profile -t nvtx,cuda -o output\_file.qdrep  
julia --project=../.. script.jl**

<https://docs.nvidia.com/nsight-visual-studio-edition/2020.1/nvtx/index.html>



# A First (I)Nsight

## Recording with the CLI

- Use the command line
  - `srun nsys profile --trace=cuda,nvtx,mpi --force-overwrite=true --output=my_report.%q{SLURM_PROCID} \ ./jacobi -niter 10`
- Inspect results: Open the report file in the GUI
  - Also possible to get details on command line
  - Either add `--stats` to profile command line, or: `nsys stats --help`
- Runs set of reports on command line, customizable (**sqlite** + **Python**):
  - Useful to check validity of profile, identify important kernels

Running [`.../reports/gpukernsum.py jacobi_metrics_more-nvtx.0.sqlite`]...

Time(%)	Total Time (ns)	Instances	Avg (ns)	Med (ns)	Min (ns)	Max (ns)	StdDev (ns)	Name
99.9	36750359	20	1837518.0	1838466.5	622945	3055044	1245121.7	void jacobi_kernel
0.1	22816	2	11408.0	11408.0	7520	15296	5498.5	initialize_boundaries

# System-level Profiling with Nsight Systems

- Global timeline view
  - CUDA HW: streams, kernels, memory
- Different traces, e.g. CUDA, MPI
  - correlations API <-> HW
- Stack samples
  - bottom-up, top-down for CPU code
- GPU metrics
- Events View
  - Expert Systems
- looks at single process (tree)
  - correlate multi-process reports in single timeline

NVIDIA Nsight Systems 2021.4.1

File View Tools Help

jacobi\_metrics\_no-nvtx.0.nsys-rep X

Timeline View

1s +850ms +900ms +950ms 2s +50ms +100ms

GPU (0000:03:00.0 - NVIDIA)

▼ CUDA HW (0000:03:00.0 -

▶ [All Streams]

▶ 61.4% Default stream 7

▶ 31.8% Stream 13

▶ 6.5% Stream 16

5 streams hidden... - +

▼ Threads (8)

▼ [10309] MPI Rank 0

MPI

CUDA API

Profiler overhead

▼ [10367] jacobi

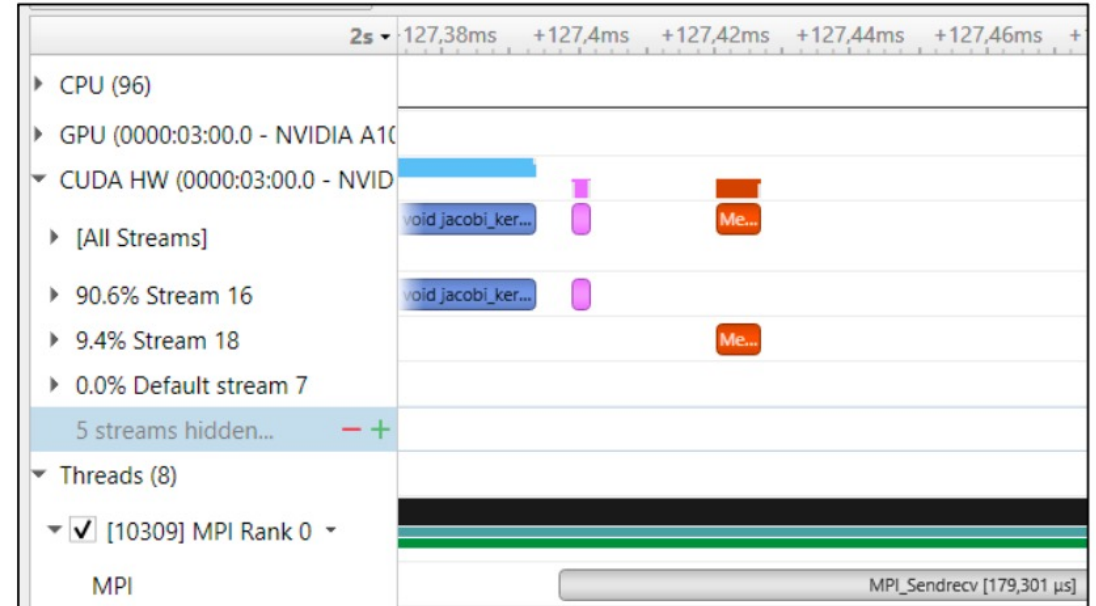
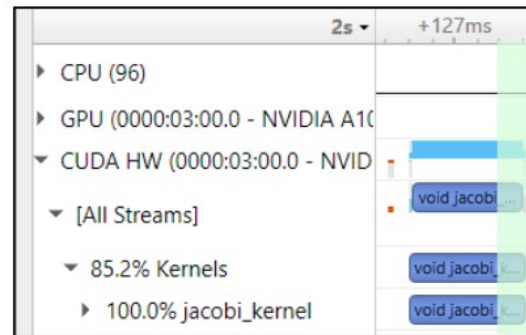
6 threads hidden... - +

Events View

#	Name	Start	Duration	TID	GPU	Context	Description:
4	Memset	1,88258s	3,200 μs	-	GPU 0	Stream 13	
5	void jacobi_ke...	1,88259s	3,056 ms	-	GPU 0	Stream 13	<b>void jacobi_kernel&lt;(int)32, (int)32&gt;(float *, const float *, float *, int, int, int, bool)</b> Begins: 1,88259s Ends: 1,88565s (+3,056 ms) grid: <<<512, 512, 1>>>
6	Memcpy DtoD	1,88565s	5,024 μs	-	GPU 0	Stream 14	
7	Memcpy DtoH	1,88565s	4,864 μs	-	GPU 0	Stream 13	

# Discovering Optimization Potential

- Using Jacobi solver example\*
- Spot kernels – lots of whitespace
  - Which part is „bad“?
  - Enhance!
- MPI calls
  - Memory copies
  - We know: This is CUDA-aware MPI
- Even without knowing source, insight
- Too complicated for repeated/reliable usage
  - How to simplify navigating and comparing reports?



\*See <https://github.com/NVIDIA/multi-gpu-programming-models/>



# Adding NVTX

## Simple range-based API

- `#include "nvtx3/nvToolsExt.h"`
  - NVTX v3 is header-only, needs just `-ldl`
  - C++ and Python APIs
- Fortran: [NVHPC compilers include module](#)
  - Just use `nvtx` and `-lnvhpcwrapnvtx`
  - Other compilers: See blog posts linked below
- Definitely: Include `PUSH/POP` macros (see links below)  
`PUSH_RANGE(name, color_idx)`
- Sprinkle them strategically through code
  - Use hierarchically: Nest ranges
- Not shown: Advanced usage (domains, ...)
- Similar range-based annotations exist for other tools
  - e.g. [SCOREP\\_USER\\_REGION\\_BEGIN](#)

```
int main(int argc, char** argv) {
    PUSH_RANGE("main", 0)
    PUSH_RANGE("init", 1)
    do_initialization();
    POP_RANGE
    /* ... */
    PUSH_RANGE("computation", 2)
    jacobi_kernel<<< /* ... */, compute_stream>>>(...);
    cudaStreamSynchronize(compute_stream);
    POP_RANGE
    /* ... */
    POP_RANGE
}
```

<https://github.com/NVIDIA/NVTX> and <https://nvidia.github.io/NVTX/#how-do-i-use-nvtx-in-my-code>

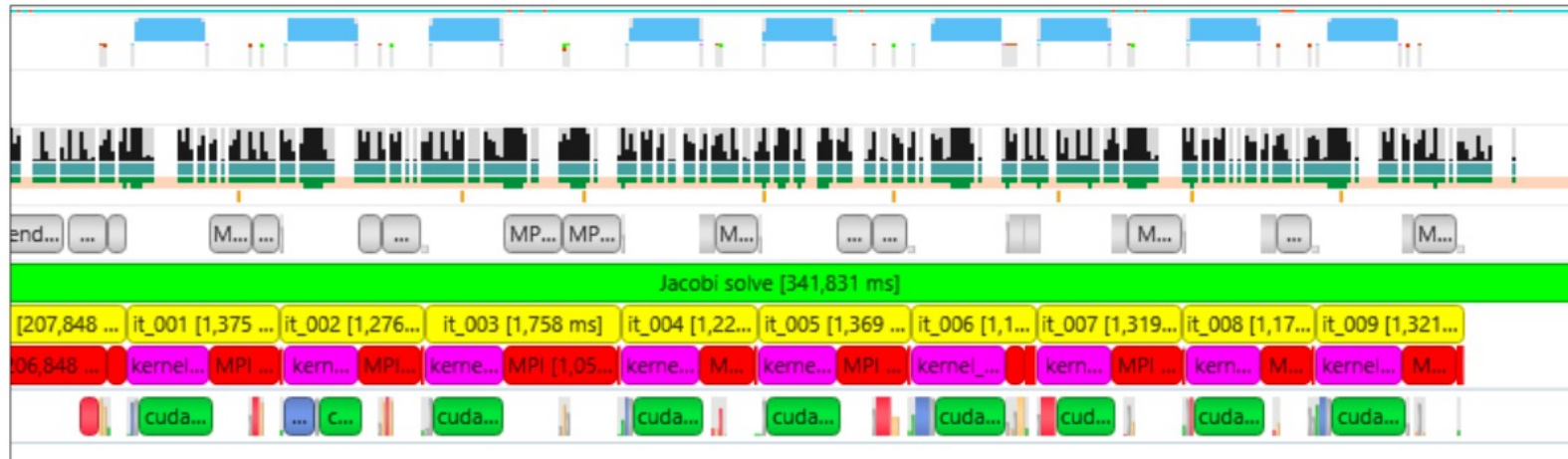
<https://developer.nvidia.com/blog/cuda-pro-tip-generate-custom-application-profile-timelines-nvtx/>  
<https://developer.nvidia.com/blog/customize-cuda-fortran-profiling-nvtx/>



# Minimizing Profile Size

Shorter time, smaller files = quicker progress

- Only profile what you need – all profilers have some overhead
  - Example: Event that occurs after long-running setup phase
- Bonus: lower number of events leads to smaller file size
- Add to nsys command line:
  - `--capture-range=nvtx --nvtx-capture=any_nvtx_marker_name \`  
`--env-var=NSYS_NVTX_PROFILER_REGISTER_ONLY=0 --kill none`
    - Use [NVTX registered strings](#) for best performance
- Alternatively: `cudaProfilerStart()` and `-Stop()`
  - `--capture-range=cudaProfilerApi`

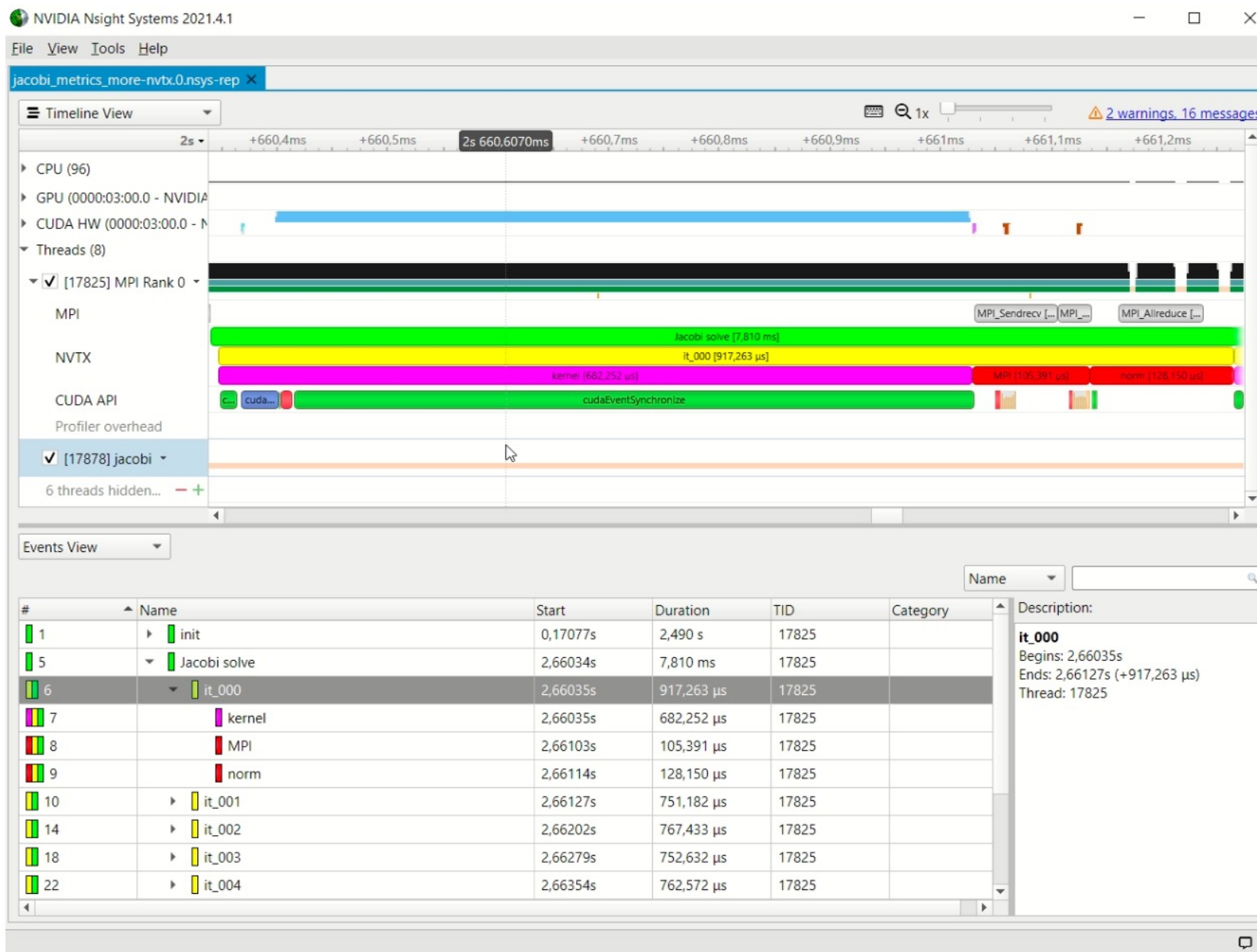


# Nsight Systems Workflow with NVTX

## Repeating the analysis



DEEP  
LEARNING  
INSTITUTE



# GPU Metrics in Nsight Systems

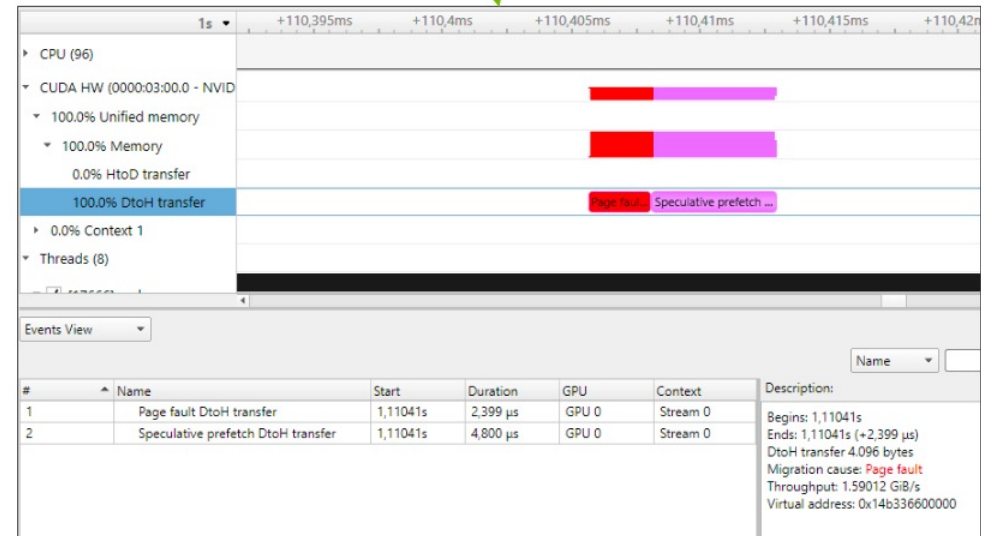
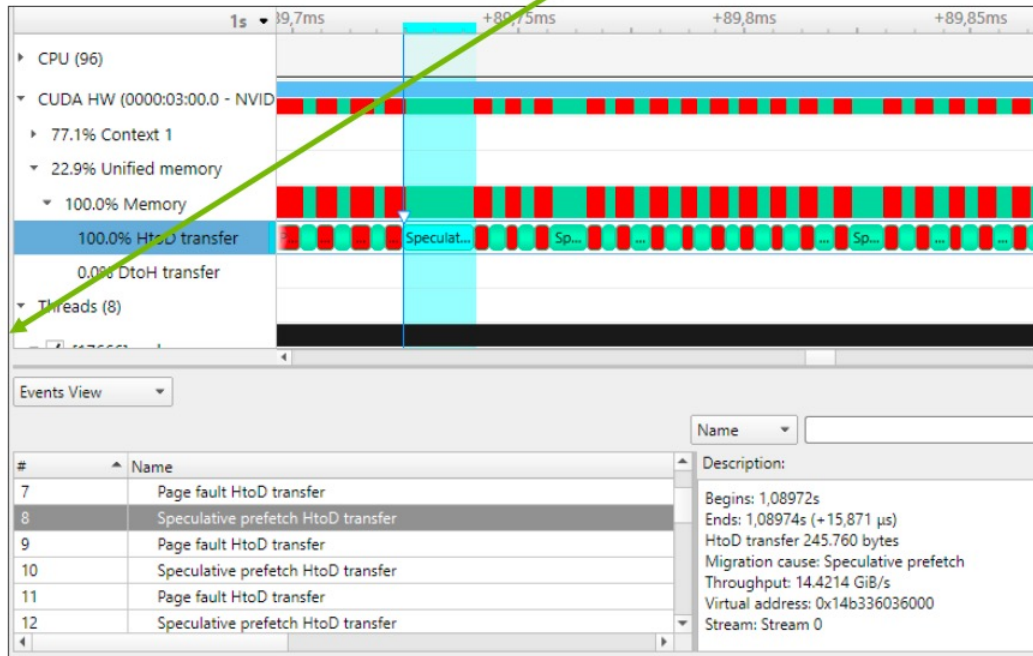
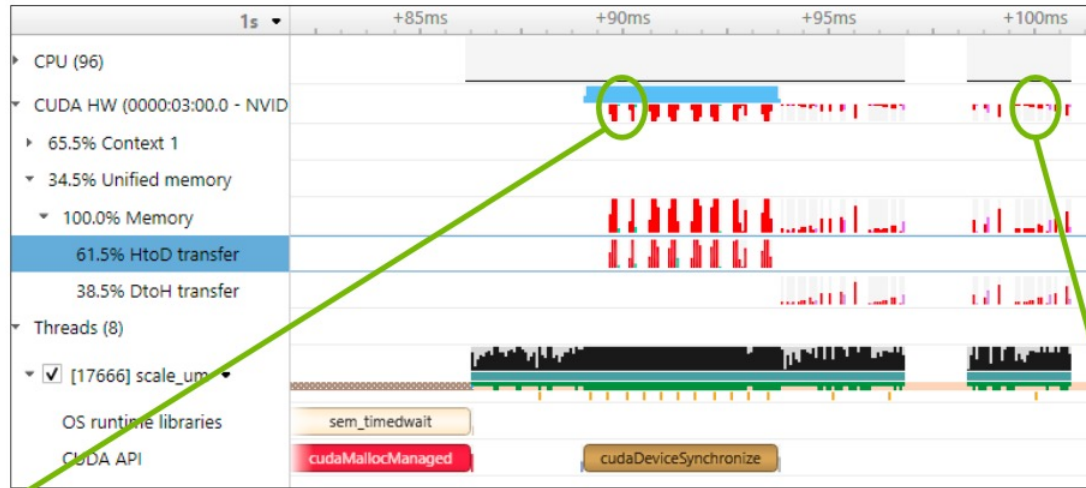
...and other traces you can activate

- Valuable low-overhead insight into HW usage:
  - SM instructions
  - DRAM Bandwidth, PCIe Bandwidth (GPUDirect)
- Also: Memory usage, Page Faults (higher overhead)
  - CUDA Programming guide: [Unified Memory Programming](#)
- Can save kernel-level profiling effort!
- `nsys profile`
  - `--gpu-metrics-device=0`
  - `--cuda-memory-usage=true`
  - `--cuda-um-cpu-page-faults=true`
  - `--cuda-um-gpu-page-faults=true`
  - `./app`

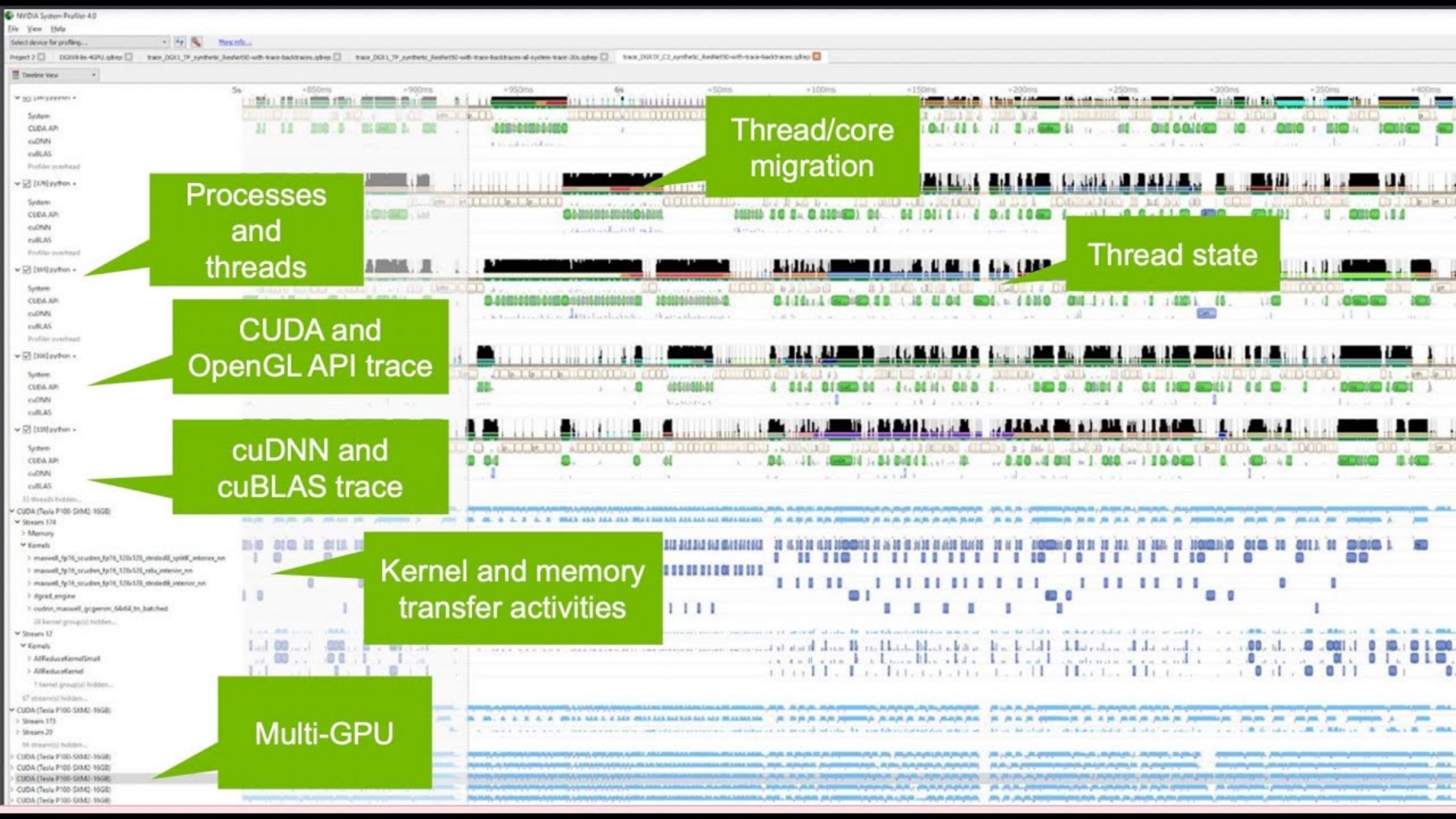


# Unified Memory movement

## Observing transfers in Nsight Systems







Processes and threads

CUDA and OpenGL API trace

cuDNN and cuBLAS trace

Thread/core migration

Thread state

Kernel and memory transfer activities

Multi-GPU





DEEP  
LEARNING  
INSTITUTE



# THANK YOU

Instructor: Dr. Momme Allalen  
[www.nvidia.com/dli](http://www.nvidia.com/dli)