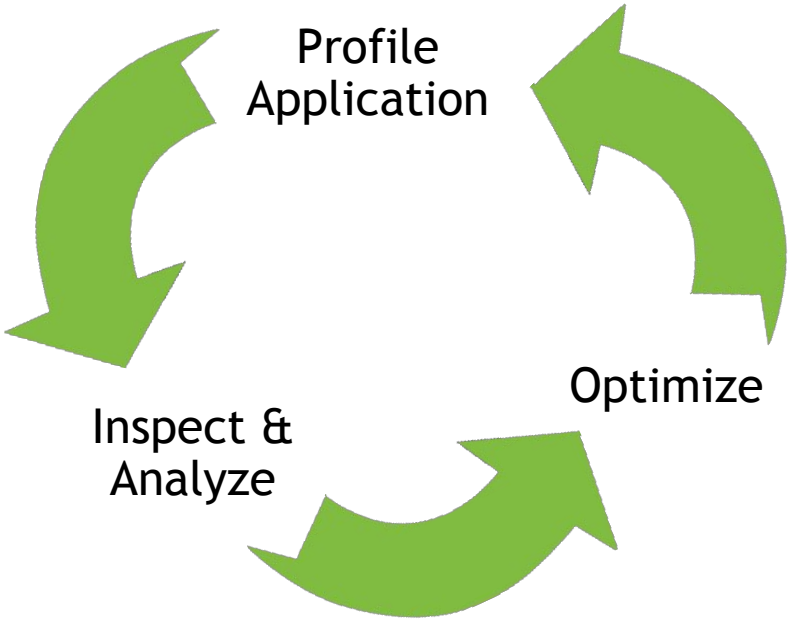
Nsight Systems - Introduction

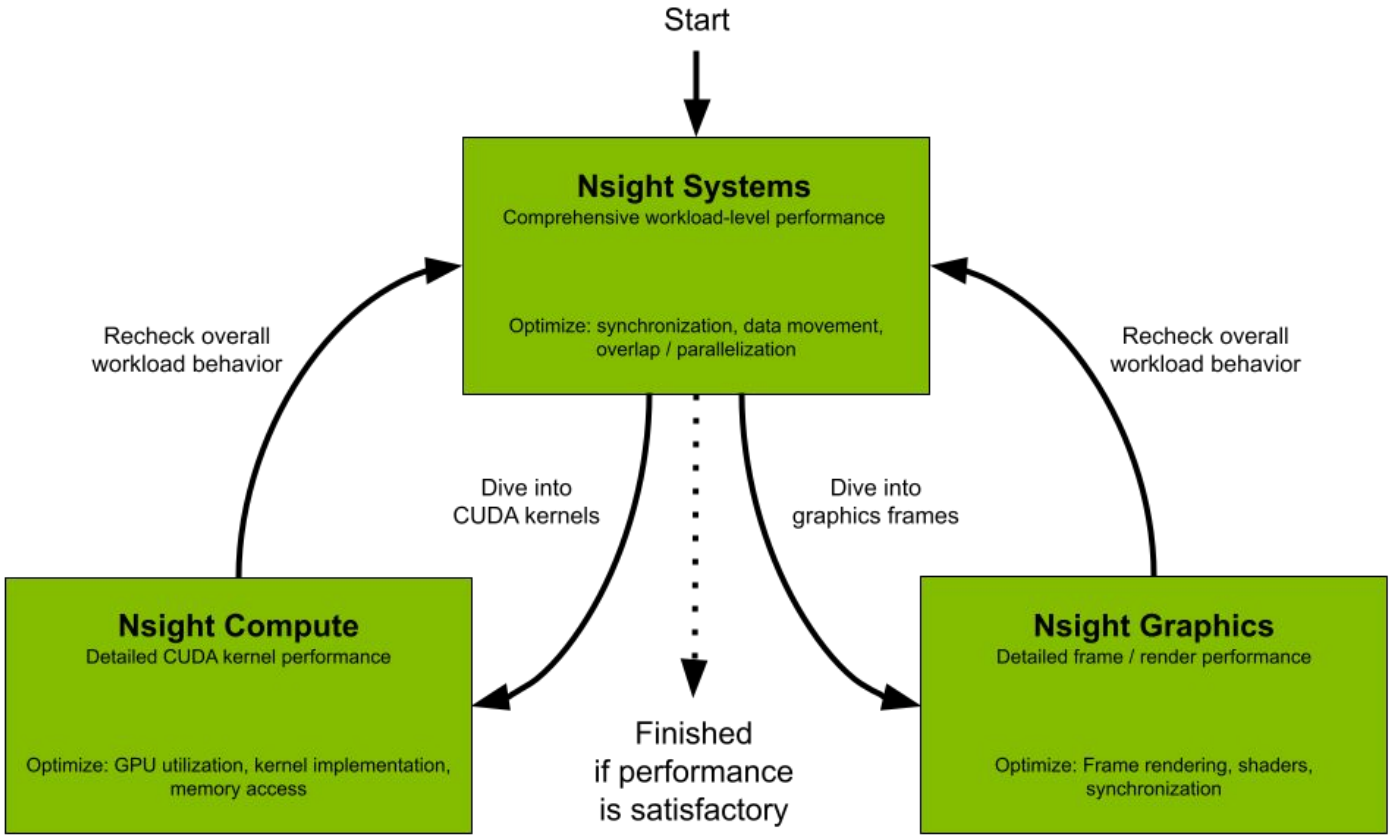
Robert Dietrich – March 17, 2022

Typical Optimization Workflow



Iterate until desired performance is achieved

NVIDIA Performance Analysis Tools



System-Wide Application Tuning

Maximize your GPU Investment

Locate optimization opportunities

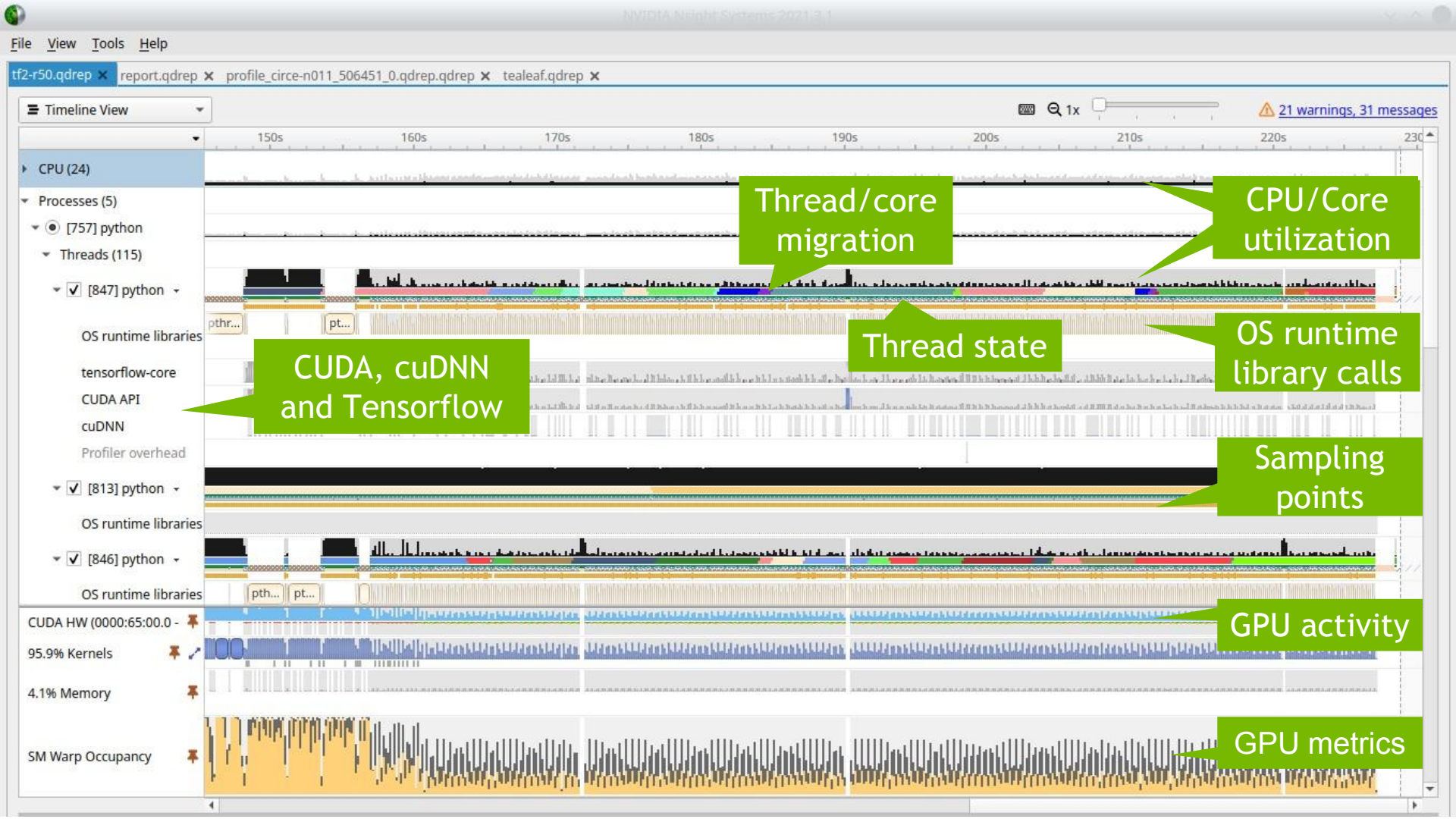
- Visualize millions of events on a timeline
- See gaps of unused CPU and GPU time

Balance your workload across multiple CPUs and GPUs

- CPU utilization and thread state
- GPU streams, kernels, memory transfers, etc.
- NIC performance metrics & tracing of network libraries

Multi-platform support

- Linux, Windows and Mac OS X (host-only)
- x86-64, Power9, ARM server, Tegra (Linux & QNX)

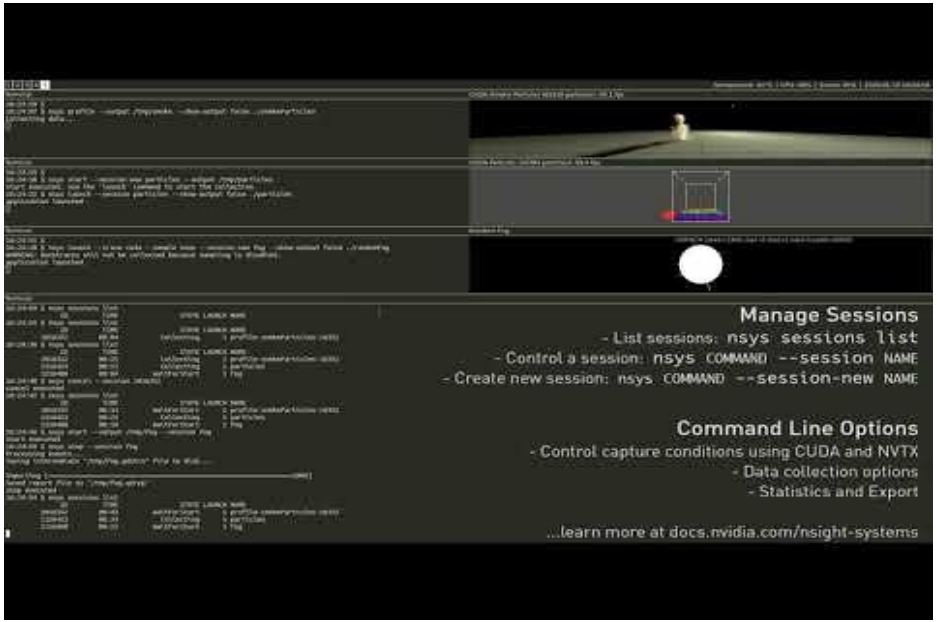

File View Tools Help
tf2-r50.qdrep
report.qdrep
profile_circe-n011_506451_0.qdrep.qdrep
tealeaf.qdrep
Timeline View
1x
21 warnings, 31 messages
150s
160s
170s
180s
190s
200s
210s
220s
CPU (24)
Processes (5)
[757] python
Threads (115)
[847] python
OS runtime libraries
tensorflow-core
CUDA API
cuDNN
Profiler overhead
[813] python
OS runtime libraries
[846] python
OS runtime libraries
CUDA HW (0000:65:00.0 -
95.9% Kernels
4.1% Memory
SM Warp Occupancy
Thread/core migration
CPU/Core utilization
Thread state
OS runtime library calls
CUDA, cuDNN and Tensorflow
Sampling points
GPU activity
GPU metrics


Command Line Interface (CLI)
Statistics and Export to SQLite, JSON, etc.

Command Line Interface

The Nsight Systems CLI provides several different commands

- Basic profiling session
  `nsys profile ./app`
- Interactive sessions (scriptable)
  `nsys start|launch|stop|cancel`
  `nsys session list`
  `nsys status|shutdown`
- Statistics and export
  `nsys stats|export`
  (export to sqlite, hdf, text, json, info)



https://youtu.be/r2ewwd4d0vc

CLI Profiling - Some Useful Switches

API tracing

```
-t, --trace=cuda,nvtx,osrt,opengl
(cublas,cusparse,cudnn,mpi,oshmem,ucx,openacc,openmp,vulkan,none)
```

Overwrite existing report

```
-f, --force-overwrite=[true|false]
```

Summary statistics (profile output on command line)

```
--stats=[true|false]
```

Report file name

```
-o, --output=report#
(patterns for hostname, PID and environment variables)
```

CLI Profiling - Some Useful Switches

Callstack sampling

`-s, --sample=[`**`cpu`**`|none]`

`--sampling-period=`*number of CPU Instructions Retired events*

`-b, --backtrace=[`**`lbr`**`|fp|dwarf|none]`

`--samples-per-backtrace={`**`1`**`..12}`

(The number of CPU IP samples collected for every CPU IP sample backtrace collected.)

Set the paranoid level: “`sudo sh -c 'echo 1 >/proc/sys/kernel/perf_event_paranoid`’

CUDA memory usage

`--cuda-memory-usage=[true|`**`false`**`]`

(Use `nsys profile --help` for a list of available options.)

CLI Profiling - MPI Programs

Single Node

```
nsys profile [nsys_args] mpirun [mpirun_args] your_executable
```

⇨ This creates one report file.

Multiple Nodes

```
mpirun [mpirun_args] nsys profile [nsys_args] your_executable
```

Set output report name with **`-o report_name_%q{OMPI_COMM_WORLD_RANK}`**
(for OpenMPI, **`PMI_RANK`** for MPICH and **`SLURM_PROCID`** for Slurm)

⇨ This creates one report file per MPI rank.

Profile only specific ranks. ⇨

```
#!/bin/bash
# OMPI_COMM_WORLD_LOCAL_RANK for node local rank
if [ $OMPI_COMM_WORLD_RANK -eq 0 ]; then
  nsys profile -t mpi "$@"
else
  "$@"
fi
```

CLI Profiling - Additional Output

```
WARNING: The command line includes a target application therefore the CPU context-switch
scope has been set to process-tree.
Collecting data…

  … APP OUTPUT …

Processing events...
Saving temporary "/tmp/nsys-report-2b96-9038-d0bd-2600.qdstrm" file to disk...

Creating final output files...
Processing [==============================================================100%]
Saved report file to "/tmp/nsys-report-2b96-9038-d0bd-2600.nsys-rep"
Report file moved to "/final/destination/report.nsys-rep"
```

Temporary data is written to **/tmp/nvidia/nsight_systems** by default. Set **TMPDIR** to specify another location.

CLI Stats Output

- CUDA API, kernels and memory operations (by time and by size)
- OS runtime
- NVTX

NVTX Push-Pop Range Statistics:

Time(%)	Total Time (ns)	Instances	Average	Minimum	Maximum	Range
80,0	16.193.376.534	21.268	761.396,0	6.823	2.438.848	MPI:MPI_Waitall
18,0	3.680.937.672	8.072	456.013,0	1.726	1.932.681	MPI:MPI_Allreduce
0,0	88.587.708	4	22.146.927,0	21.714.183	22.579.671	MPI:MPI_Init
0,0	57.888.066	21.268	2.721,0	1.319	297.466	MPI:MPI_Isend
0,0	24.282.234	21.268	1.141,0	872	16.931	MPI:MPI_Irecv
0,0	8.793.048	4	2.198.262,0	2.181.292	2.215.232	MPI:MPI_Finalize
0,0	3.814.158	28	136.219,0	2.302	1.265.786	MPI:MPI_Barrier
0,0	266.630	32	8.332,0	832	104.980	MPI:MPI_Reduce

CUDA Kernel Statistics:

Time(%)	Total Time (ns)	Instances	Average	Minimum	Maximum	Name
37,0	7.344.350.347	8.568	857.183,0	851.546	876.922	device_tea_leaf_ppcg_solve_calc_sd_new(kernel_info_t, dou
36,0	7.229.217.310	8.568	843.746,0	839.323	899.482	device_tea_leaf_ppcg_solve_update_r(kernel_info_t, double
10,0	2.050.961.573	2.010	1.020.378,0	1.005.849	1.036.569	device_tea_leaf_cg_solve_calc_ur(kernel_info_t, double, d
9,0	1.879.919.365	2.010	935.283,0	913.754	1.214.808	device_tea_leaf_cg_solve_calc_w(kernel_info_t, double*, d
5,0	1.008.596.179	1.980	509.392,0	501.469	521.756	device_tea_leaf_cg_solve_calc_p(kernel_info_t, double, do
0,0	61.606.046	30	2.053.534,0	1.176.600	2.890.541	device_tea_leaf_calc_rrn(kernel_info_t, double const*, do
0,0	38.094.843	12.336	3.088,0	1.887	13.184	void reduction<double, (REDUCTION_TYPE)0>(int, double)


NVTX
NVIDIA Tool Extension Library

NVTX

NVIDIA Tool Extension Library

NVTX provides means to correlate the profile data with the application code.

```
#include <nvtx3/nvToolsExt.h>
…
nvtxMark("Point in time");
…
nvtxRangePush("Name of your code region");
// your code goes here
nvtxRangePop();
…
nvtxRangeId_t rid = nvtxRangeStart("Name of another code region");
// stop might be on another thread
nvtxRangeEnd(rid)
```

(Add **nvtx** to the tracing options of **nsys profile**.)

NVTX

Domains and Resource Naming

```
nvtxDomainHandle_t yourDomain = nvtxDomainCreateA("My Domain");

nvtxEventAttributes_t evtAttr = {0};
evtAttr.version = NVTX_VERSION;
evtAttr.size = NVTX_EVENT_ATTRIB_STRUCT_SIZE;
evtAttr.messageType = NVTX_MESSAGE_TYPE_ASCII;
eventAttrib.message.ascii = "My Range";

nvtxDomainRangePushEx(yourDomain, &evtAttr);
// your code
nvtxDomainRangePop(yourDomain);


// #include <sys/syscall.h>
nvtxNameOsThreadA(syscall(SYS_gettid), "My Thread");
```

NVTX

Registered Strings

```
nvtxDomainHandle_t yourDomain = nvtxDomainCreateA("My Domain");
nvtxStringHandle_t yourStrHdl =
     nvtxDomainRegisterStringA(yourDomain, "My String Handle");

nvtxEventAttributes_t evtAttr = {0};
evtAttr.version = NVTX_VERSION;
evtAttr.size = NVTX_EVENT_ATTRIB_STRUCT_SIZE;
evtAttr.messageType = NVTX_MESSAGE_TYPE_REGISTERED;
eventAttrib.message.registered = yourStrHdl;

nvtxDomainRangePushEx(yourDomain, &evtAttr);
// your code
nvtxDomainRangePop(yourDomain);
```

https://github.com/NVIDIA/NVTX

NVTX

Fortran Bindings

The NVIDIA HPC SDK Fortran compiler provides NVTX bindings.

- **`libnvhpcwrapnvtx.[a|so]`** has to be linked
- Documentation can be found here: https://docs.nvidia.com/hpc-sdk/compilers/fortran-cuda-interfaces/index.html#cfnvtx-runtime

```
use nvtx
…
call
nvtxStartRange("YourRange")
! some Fortran code
call nvtxEndRange
```

For other compilers, write your own Fortran NVTX bindings or use existing ones, e.g. https://raw.githubusercontent.com/maxcuda/NVTX_example/master/nvtx.f90

NVTX

Python Bindings

“Python developers can either use decorators `@nvtx.annotate()` or a context manager with `nvtx.annotate(..)`”

```
import nvtx

@nvtx.annotate(“f()”, color="purple")
def f():
    for i in range(5):
        with nvtx.annotate("loop", color="red"):
            # Python code goes here
```

Get NVTX Python module:

```
python -m pip install nvtx
```

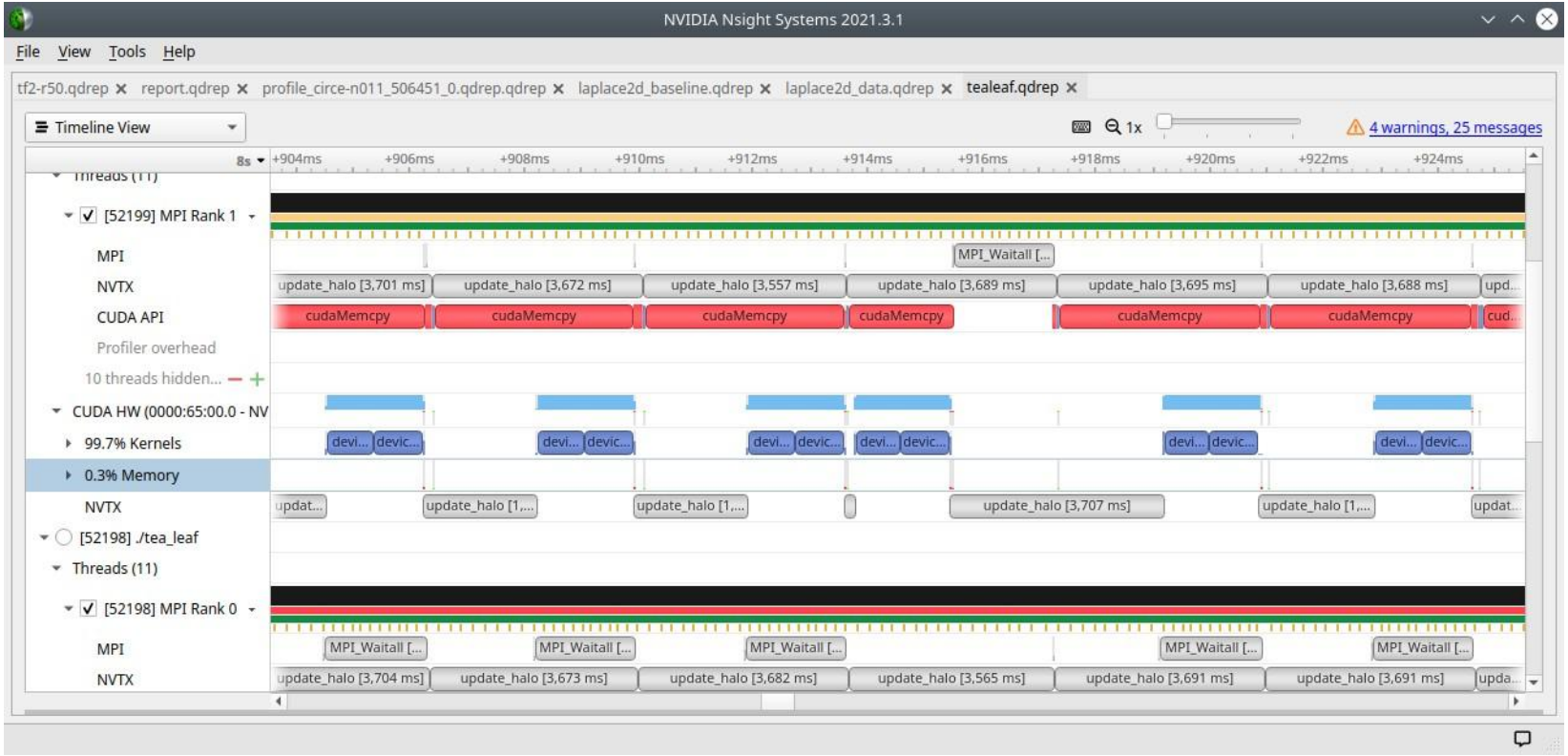
Also see this blog or the docs: https://nvtx.readthedocs.io/en/latest/index.html

PyTorch CUDA provides NVTX bindings

```
from torch.cuda import nvtx

nvtx.range_push("YourCode")
# your Python code
nvtx.range_pop()
```


Demo: Teafleaf

TeaLeaf Demo



Sample from https://github.com/UK-MAC/TeaLeaf

Build TeaLeaf

```
# Get the source code
git clone https://github.com/UK-MAC/TeaLeaf_CUDA.git ; cd TeaLeaf_CUDA


# Annotate the code
```

```
use nvtx
…
call nvtxStartRange("Update Halo")
! some fortran code
call nvtxEndRange
```

```
# Setup environment (MPI, CUDA, compiler, etc.)

# Build TeaLeaf
#   add AMPERE architecture and -std=c++14 flag for nvcc
make -e COMPILER=PGI
```

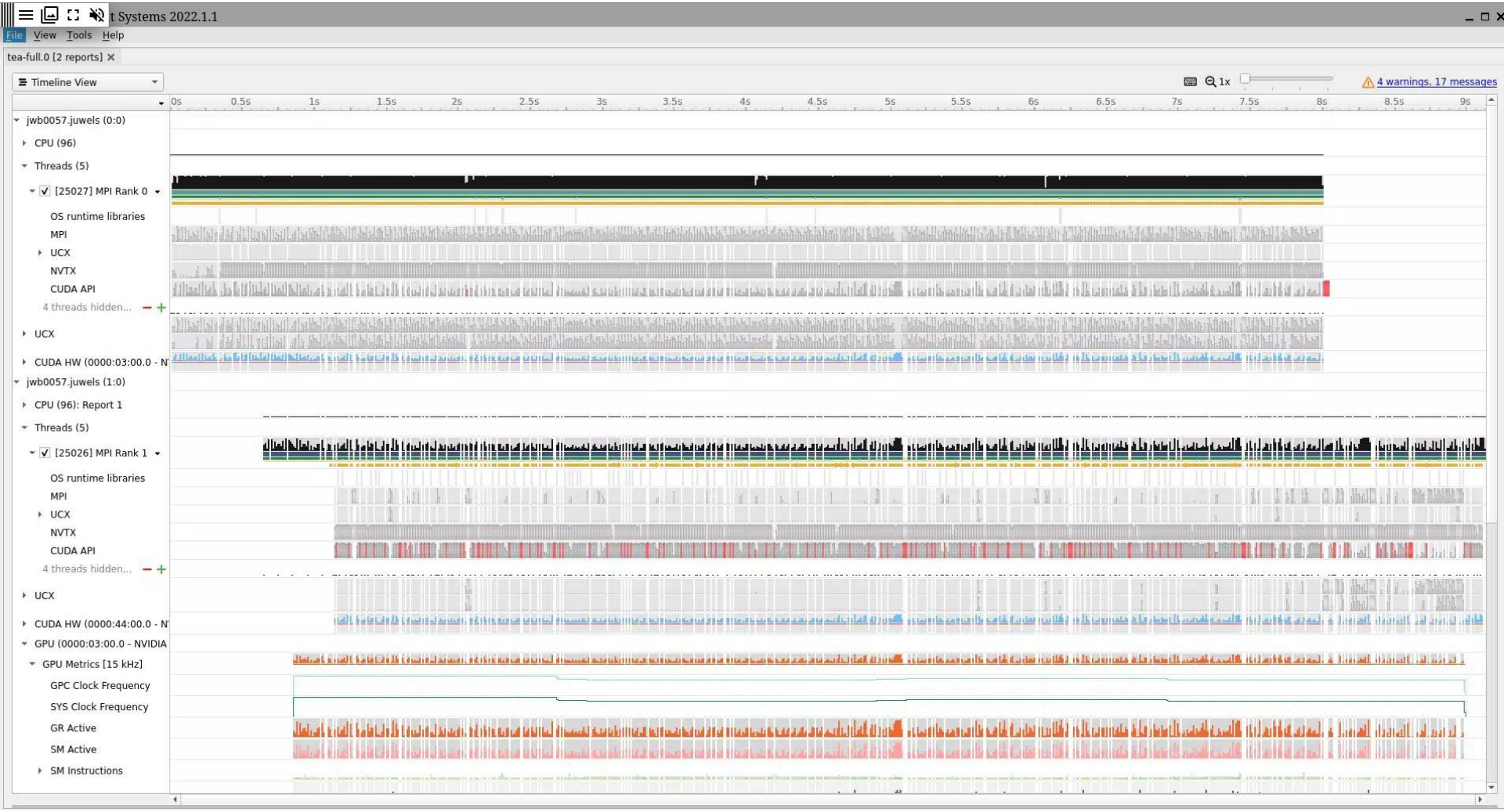
Profile TeaLeaf

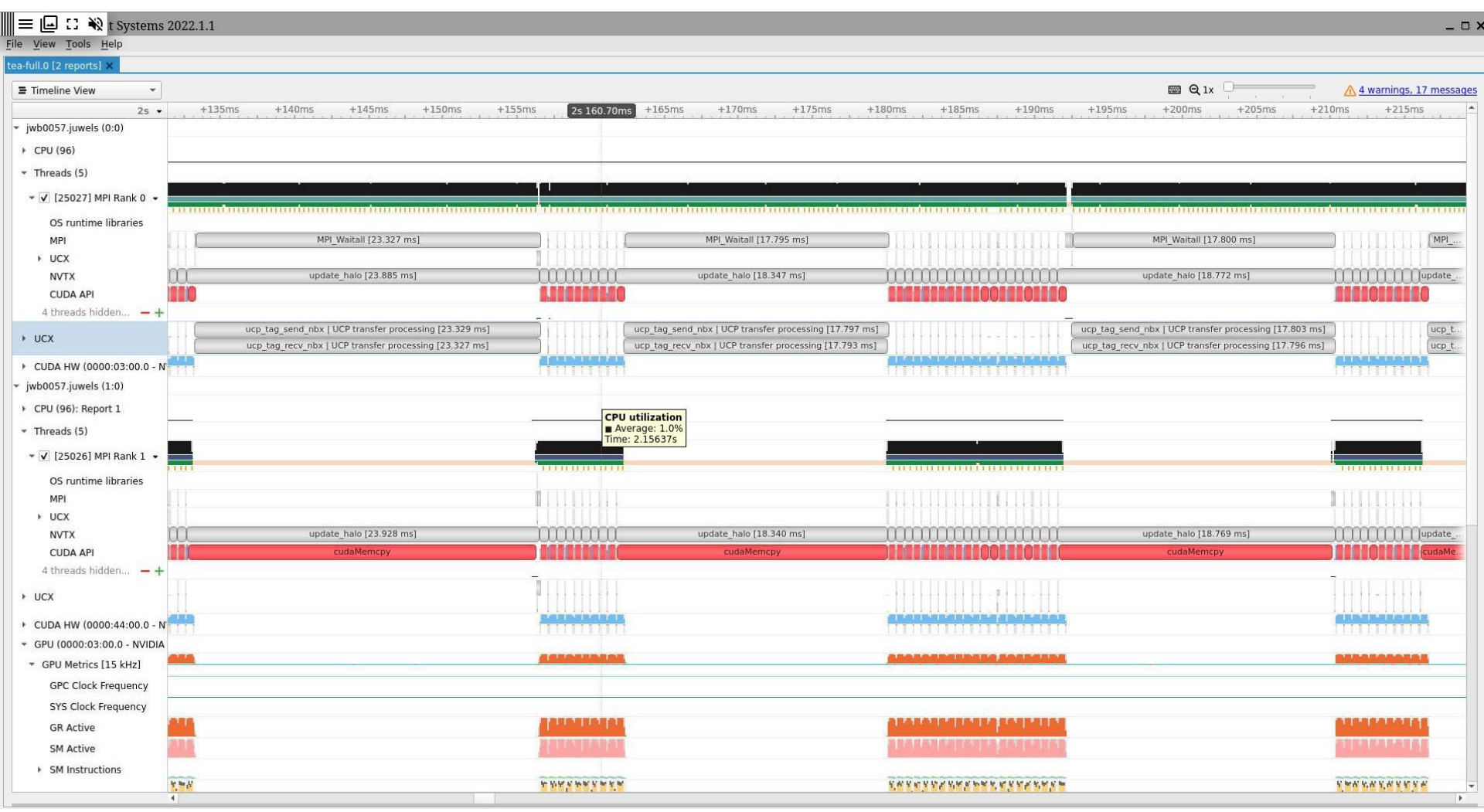
```
# Run TeaLeaf without profiling
srun/mpirun [args] ./tea_leaf

# Profile with NVTX
srun/mpirun [args] \
nsys profile -t cuda,mpi,ucx,osrt,nvtx \
-y 1 -d 8 --kill=none \
-o tea-nvtx.%q{SLURM_PROCID} --stats=true \
./tea_leaf


# Profile with CPU and GPU sampling
srun/mpirun [args] \
nsys profile -t cuda,mpi,ucx,osrt,nvtx \
-y 1 -d 8 --kill=none -o tea-sampling%q{SLURM_PROCID} \
--backtrace=dwarf --sampling-period=3000000 \
--gpu-metrics-set=ga100 --gpu-metrics-device=0,1 \
--gpu-metrics-frequency=15000 \
./tea_leaf
```

Analyze the Profile

Investigate CLI stats output (`--stats`)

Open the report file in the Nsight Systems GUI: `nsys-ui tea-nvtx.0.nsys-rep`


t Systems 2022.1.1
File View Tools Help
tea-full.0 [2 reports]
Timeline View
1x
4 warnings, 17 messages
0s 0.5s 1s 1.5s 2s 2.5s 3s 3.5s 4s 4.5s 5s 5.5s 6s 6.5s 7s 7.5s 8s 8.5s 9s
jwb0057.juwels (0:0)
CPU (96)
Threads (5)
[25027] MPI Rank 0
OS runtime libraries
MPI
UCX
NVTX
CUDA API
4 threads hidden...
UCX
CUDA HW (0000:03:00.0 - N
jwb0057.juwels (1:0)
CPU (96): Report 1
Threads (5)
[25026] MPI Rank 1
OS runtime libraries
MPI
UCX
NVTX
CUDA API
4 threads hidden...
UCX
CUDA HW (0000:44:00.0 - N
GPU (0000:03:00.0 - NVIDIA
GPU Metrics [15 kHz]
GPC Clock Frequency
SYS Clock Frequency
GR Active
SM Active
SM Instructions


t Systems 2022.1.1
File View Tools Help
tea-full.0 [2 reports]
Timeline View
1x
4 warnings, 17 messages
2s
+135ms
+140ms
+145ms
+150ms
+155ms
2s 160.70ms
+165ms
+170ms
+175ms
+180ms
+185ms
+190ms
+195ms
+200ms
+205ms
+210ms
+215ms
jwb0057.juwels (0:0)
CPU (96)
Threads (5)
[25027] MPI Rank 0
OS runtime libraries
MPI
UCX
NVTX
CUDA API
4 threads hidden...
MPI_Waitall [23.327 ms]
MPI_Waitall [17.795 ms]
MPI_Waitall [17.800 ms]
update_halo [23.885 ms]
update_halo [18.347 ms]
update_halo [18.772 ms]
UCX
ucp_tag_send_nbx | UCP transfer processing [23.329 ms]
ucp_tag_recv_nbx | UCP transfer processing [23.327 ms]
ucp_tag_send_nbx | UCP transfer processing [17.797 ms]
ucp_tag_recv_nbx | UCP transfer processing [17.793 ms]
ucp_tag_send_nbx | UCP transfer processing [17.803 ms]
ucp_tag_recv_nbx | UCP transfer processing [17.796 ms]
CUDA HW (0000:03:00.0 - N
jwb0057.juwels (1:0)
CPU (96): Report 1
CPU utilization
Average: 1.0%
Time: 2.15637s
Threads (5)
[25026] MPI Rank 1
OS runtime libraries
MPI
UCX
NVTX
CUDA API
4 threads hidden...
update_halo [23.928 ms]
update_halo [18.340 ms]
update_halo [18.769 ms]
cudaMemcpy
cudaMemcpy
cudaMemcpy
UCX
CUDA HW (0000:44:00.0 - N
GPU (0000:03:00.0 - NVIDIA
GPU Metrics [15 kHz]
GPC Clock Frequency
SYS Clock Frequency
GR Active
SM Active
SM Instructions

t Systems 2022.1.1

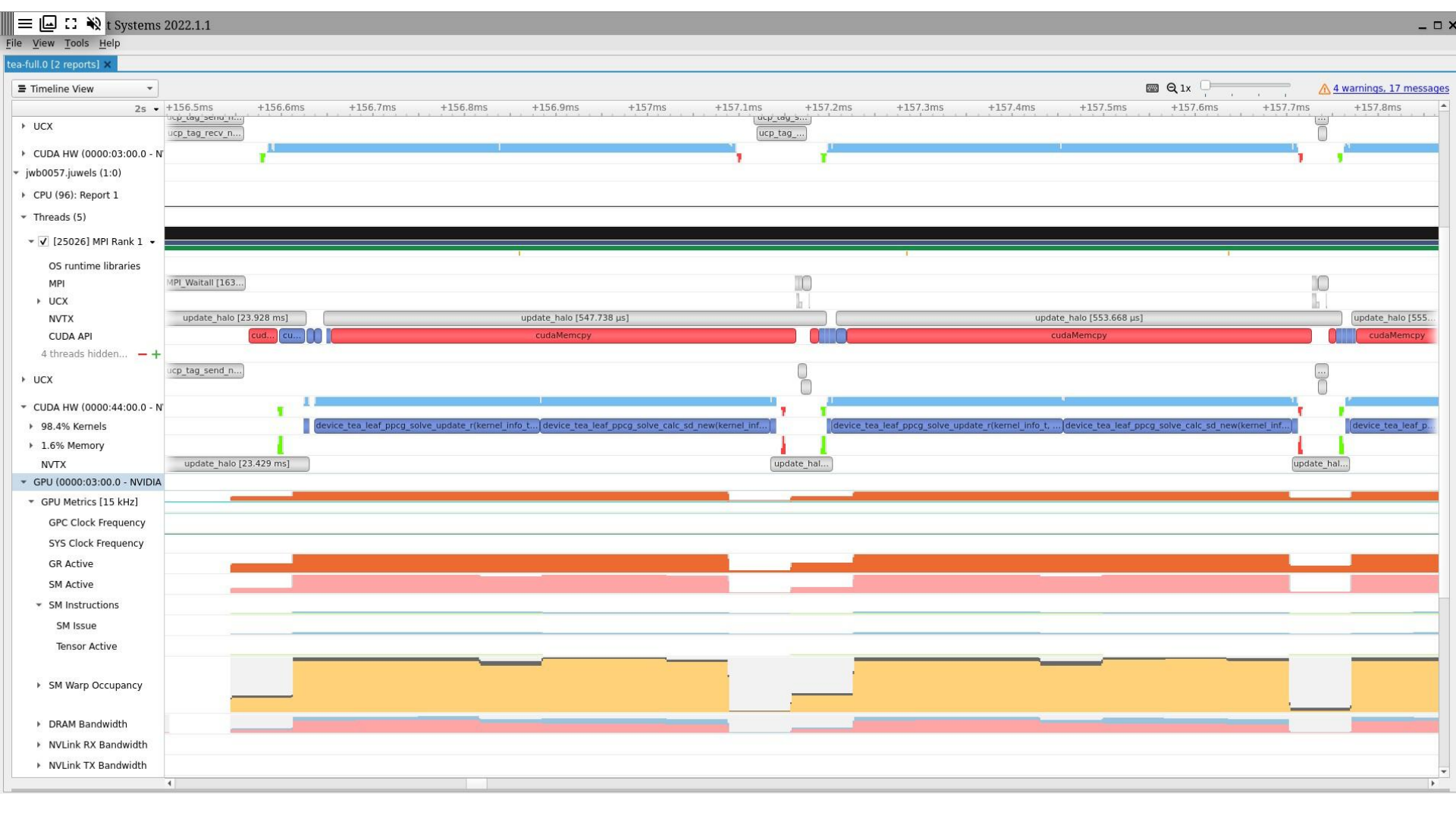
File View Tools Help

tea-full.0 [2 reports]

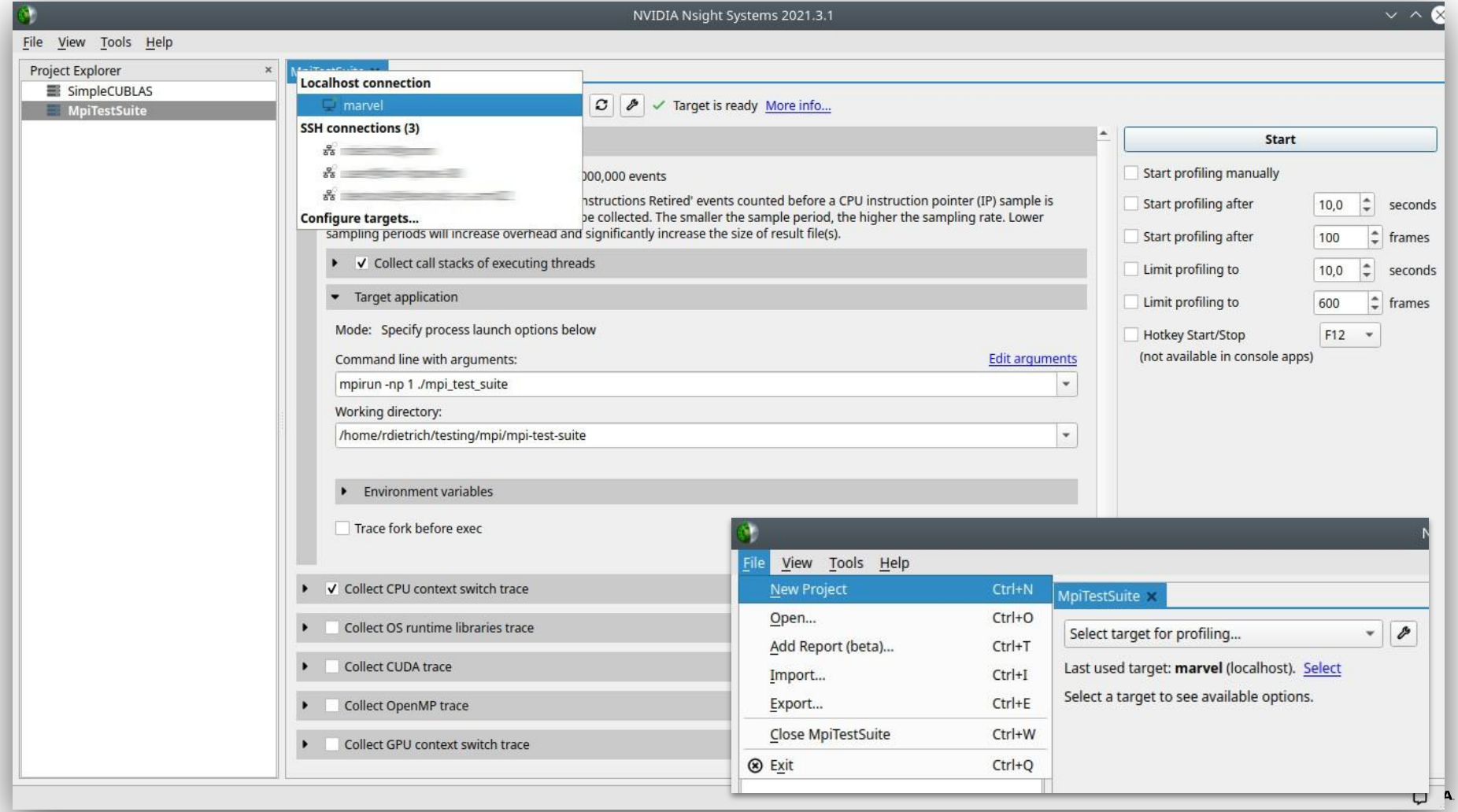
Diagnostics Summary | Report: 1

Messages

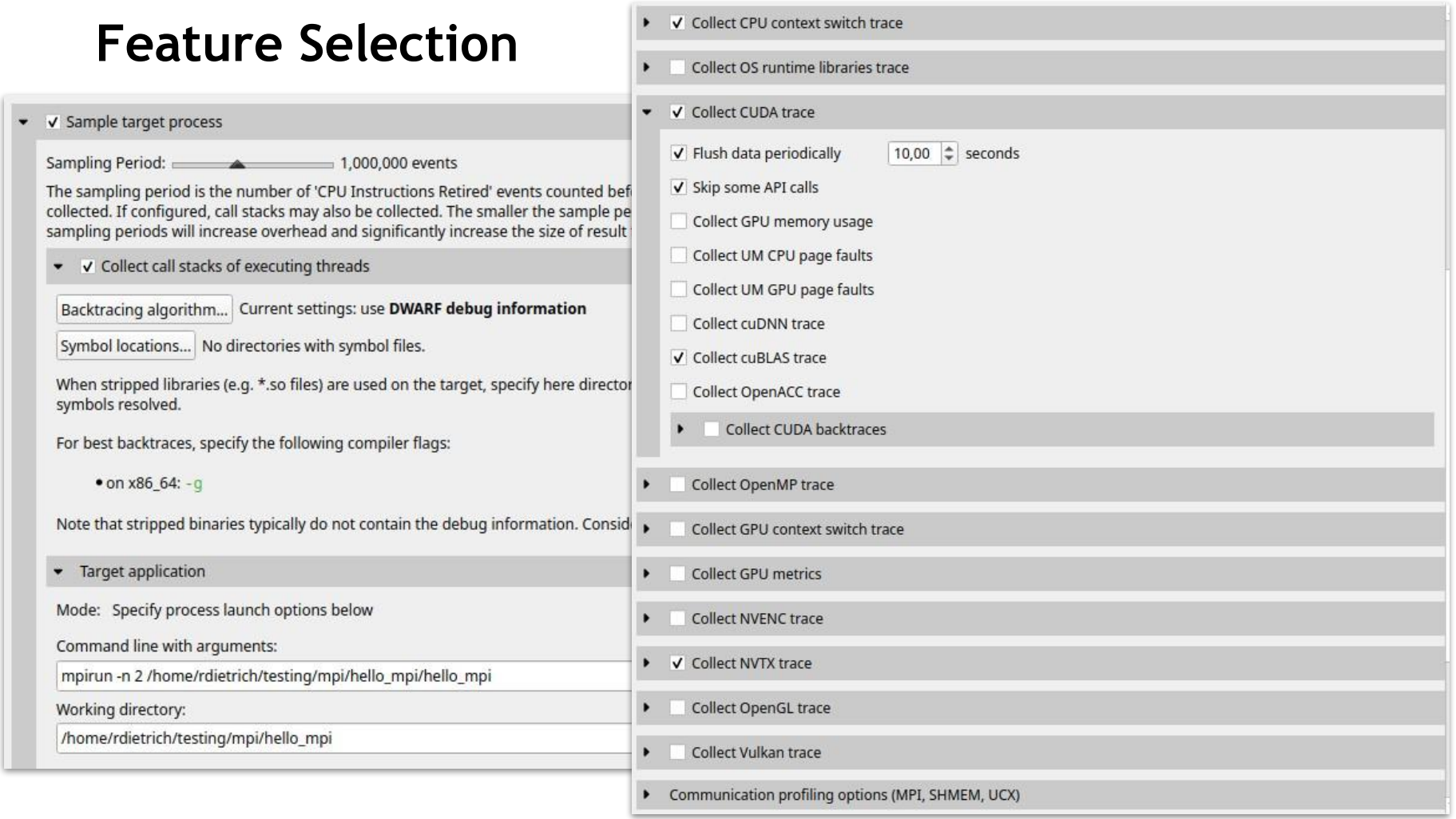
	Source	Process ID	Time	Description
ℹ	Injection	25026	-00:00.766	OS runtime libraries injection initialized successfully.
ℹ	Injection	25026	-00:00.766	NVTX injection initialized successfully.
ℹ	Injection	25026	-00:00.766	Buffers holding CUDA trace data will be flushed on CudaProfilerStop() call.
ℹ	Injection	25026	-00:00.766	CUDA injection initialized successfully.
ℹ	Injection	25026	-00:00.766	Common injection library initialized successfully.
⚠	Daemon		-00:00.428	Unable to collect CPU kernel IP samples and backtraces. perf event paranoid level is 2. Change the paranoid level to 1 to enable kernel sample collection. Try sudo sh -c 'echo 1 >/proc/sys/kernel/perf_event_paranoid' to change the paranoid level to 1.
ℹ	Daemon		-00:00.412	Dwarf backtraces collected.
ℹ	Daemon		-00:00.412	Hardware event 'instructions', with sampling period 3000000, used to trigger sample collection.
ℹ	Daemon		00:00.640	4 CPU IP samples collected for every CPU IP backtrace collected.
ℹ	Analysis		00:01.110	Profiling has started.
ℹ	Analysis		00:01.412	Number of GPU Metrics events collected: 199,298.
⚠	Analysis	25026	00:01.874	Not all NVTX events might have been collected.
ℹ	Analysis	25026	00:01.874	Number of NVTX events collected: 40,871.
⚠	Analysis	25026	00:01.874	Not all CUDA events might have been collected.
ℹ	Analysis	25026	00:01.874	Number of CUDA events collected: 55,019.
⚠	Analysis	25026	00:01.874	Not all OS runtime libraries events might have been collected.
ℹ	Analysis	25026	00:01.874	Number of OS runtime libraries events collected: 204.
ℹ	Analysis		00:09.134	Profiling has stopped.
ℹ	Injection	25026	00:09.167	Number of CUPTI events produced: 55,331, CUPTI buffers: 50.
ℹ	Daemon		00:09.793	Number of IP samples collected: 7,894.
ℹ	Daemon	25026	08:27:01.988	Process was launched by the profiler, see /tmp/nvidia/nsight_systems/quadd_session_1125009/streams/pid_25026_stdout.log and stderr.log for program output


t Systems 2022.1.1
File View Tools Help
tea-full.0 [2 reports]
Timeline View
1x
4 warnings, 17 messages
2s
+156.5ms
+156.6ms
+156.7ms
+156.8ms
+156.9ms
+157ms
+157.1ms
+157.2ms
+157.3ms
+157.4ms
+157.5ms
+157.6ms
+157.7ms
+157.8ms
UCX
ucp_tag_recv_n...
ucp_tag_...
CUDA HW (0000:03:00.0 - N
jwb0057.juwels (1:0)
CPU (96): Report 1
Threads (5)
[25026] MPI Rank 1
OS runtime libraries
MPI
MPI_Waitall [163...
UCX
NVTX
update_halo [23.928 ms]
update_halo [547.738 µs]
update_halo [553.668 µs]
update_halo [555...
CUDA API
cud...
cu...
cudaMemcpy
cudaMemcpy
cudaMemcpy
4 threads hidden...
UCX
ucp_tag_send_n...
CUDA HW (0000:44:00.0 - N
98.4% Kernels
1.6% Memory
NVTX
device_tea_leaf_ppcg_solve_update_r(kernel_info_t...
device_tea_leaf_ppcg_solve_calc_sd_new(kernel_inf...
device_tea_leaf_ppcg_solve_update_r(kernel_info_t, ...
device_tea_leaf_ppcg_solve_calc_sd_new(kernel_inf...
device_tea_leaf_p...
update_halo [23.429 ms]
update_hal...
update_hal...
GPU (0000:03:00.0 - NVIDIA
GPU Metrics [15 kHz]
GPC Clock Frequency
SYS Clock Frequency
GR Active
SM Active
SM Instructions
SM Issue
Tensor Active
SM Warp Occupancy
DRAM Bandwidth
NVLink RX Bandwidth
NVLink TX Bandwidth


Launch Application via GUI


NVIDIA Nsight Systems 2021.3.1
File View Tools Help
Project Explorer
SimpleCUBLAS
MpiTestSuite
Localhost connection
marvel
SSH connections (3)
Configure targets...
Target is ready More info...
Start
Start profiling manually
Start profiling after 10,0 seconds
Start profiling after 100 frames
Limit profiling to 10,0 seconds
Limit profiling to 600 frames
Hotkey Start/Stop F12
(not available in console apps)
000,000 events
nstructions Retired' events counted before a CPU instruction pointer (IP) sample is
e collected. The smaller the sample period, the higher the sampling rate. Lower
sampling periods will increase overhead and significantly increase the size of result file(s).
Collect call stacks of executing threads
Target application
Mode: Specify process launch options below
Command line with arguments:
Edit arguments
mpirun -np 1 ./mpi_test_suite
Working directory:
/home/rdietrich/testing/mpi/mpi-test-suite
Environment variables
Trace fork before exec
Collect CPU context switch trace
Collect OS runtime libraries trace
Collect CUDA trace
Collect OpenMP trace
Collect GPU context switch trace
File View Tools Help
New Project Ctrl+N
Open... Ctrl+O
Add Report (beta)... Ctrl+T
Import... Ctrl+I
Export... Ctrl+E
Close MpiTestSuite Ctrl+W
Exit Ctrl+Q
MpiTestSuite
Select target for profiling...
Last used target: marvel (localhost). Select
Select a target to see available options.

Feature Selection



Nsight Systems - Summary

Today

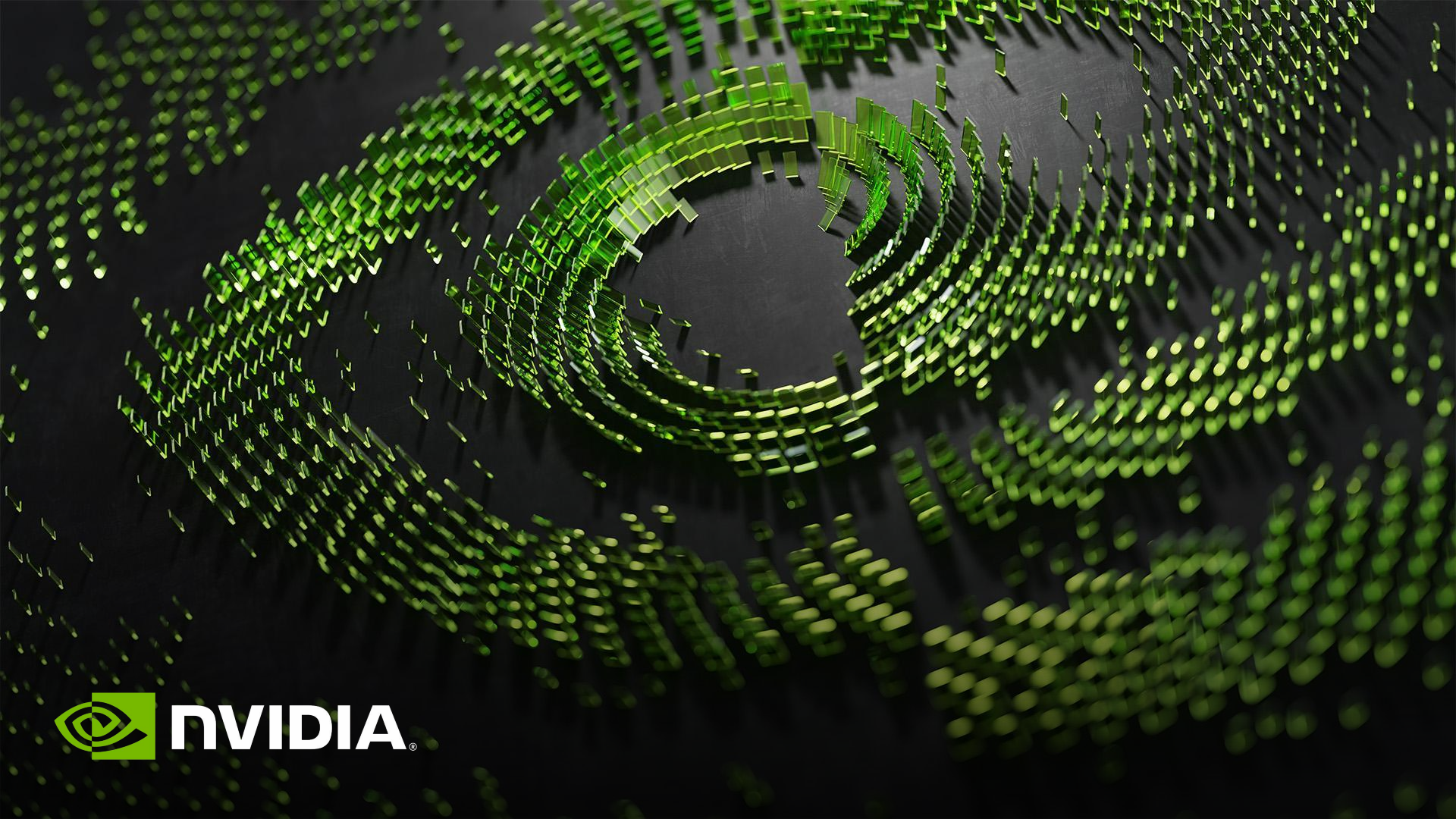
- Mainly used for single server profiling runs
- Focused on GPU workloads, API trace, thread samples
- Cluster profiling
  - Load multiple reports into one timeline
  - MPI (including communication parameters), OpenSHMEM, UCX tracing
  - NIC metrics sampling (congestions, read/write bandwidth)

Roadmap

- Improve multi-node support, e.g. correlation across nodes, switch statistics, etc.
- Determine latency, congestion, hot devices (NICs, switches, network storage)

THANK YOU!

Download	https://developer.nvidia.com/nsight-systems NOTE: website version is newer than CUDA Toolkit version
Docs	https://docs.nvidia.com/nsight-systems/index.html
Forums	https://devtalk.nvidia.com
Email	nsight-systems@nvidia.com
Blogs	• https://developer.nvidia.com/blog/nvidia-nsight-systems-containers-cloud • https://developer.nvidia.com/blog/nsight-systems-exposes-gpu-optimization • https://developer.nvidia.com/blog/understanding-the-visualization-of-overhead-and-latency-in-nsight-systems • https://developer.nvidia.com/blog/nvidia-tools-extension-api-nvtx-annotation-tool-for-profiling-code-in-python-and-c-c/


NVIDIA