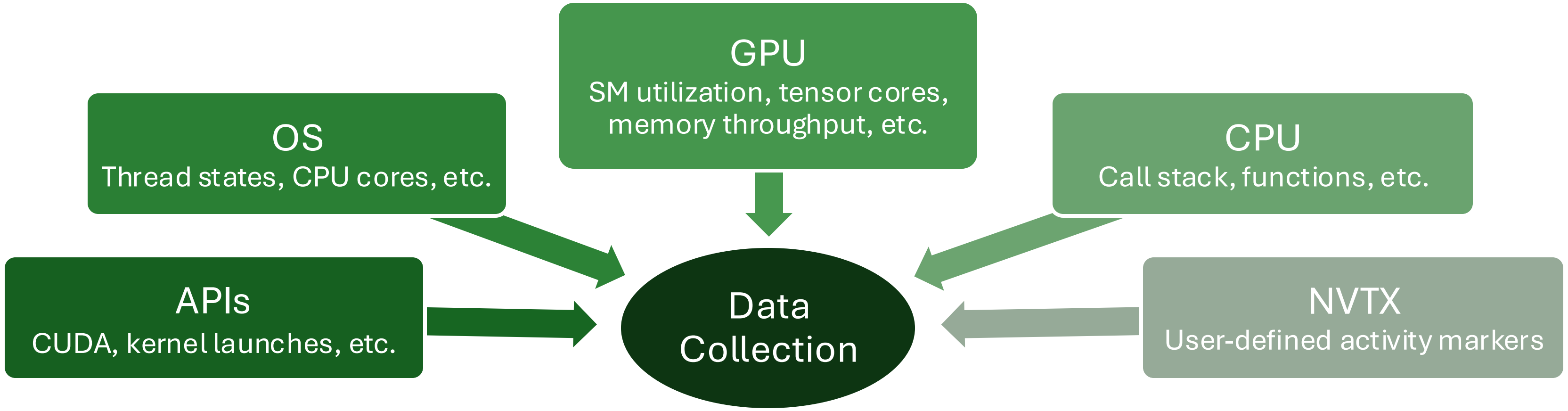
An Intro to Nvidia Nsight Systems

Nov 14 2025

How does it work?

About

- Nsight Systems is a system-wide performance analysis, we can capture activity across the CPU, GPU, and OS!
- Uses low-overhead tracing/sampling to find bottlenecks – thus can't exactly tell you why slowdowns are occurring, just where + when.


GPU
SM utilization, tensor cores, memory throughput, etc.
OS
Thread states, CPU cores, etc.
CPU
Call stack, functions, etc.
APIs
CUDA, kernel launches, etc.
Data Collection
NVTX
User-defined activity markers

Optimization

- If bottleneck is GPU-bound, we can use Nsight Compute to analyze specific kernels (not discussed here)
- Otherwise, profile as normal.

https://docs.nvidia.com/nsight-systems/UserGuide/index.html , https://docs.nvidia.com/nsight-systems/AnalysisGuide/index.html

Installation

- Download from here: https://developer.nvidia.com/nsight-systems/get-started#reqs
  - Needs to be on both your workstation and personal device! (if you are using SSH)
- Set execution privileges in the file via chmod +x NVIDIA_Nsight_Systems_Linux_*.run
- Run ./NVIDIA_Nsight_Systems_Linux_*.run to make it available

Profiling

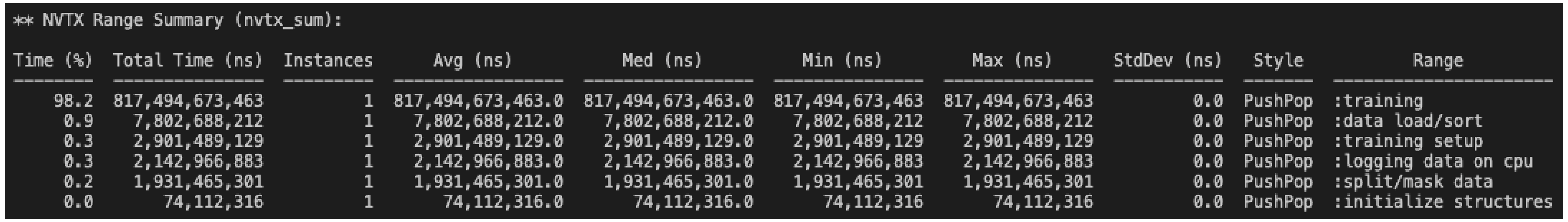
- **Post-run analysis** (most common/easier to use)
  - Run nsys profile -o report julia script.jl
    - ”report”: name of the output file, ”script”: name of the input file
- **During-run analysis** (not discussed here)
  - Interactive CLI
    - Launch in terminal w/ manual control over what runs
    - Run nsys launch julia –project=.script.jl, nsys start/stop, etc.
  - Interactive GUI
    - Launch in program on remote and view locally as it runs
    - Run ssh -X user@server nsys-ui
- Each method outputs a .nsys-rep file which we can analyze further...

Flag	Description
-o, --output <name>	Filename for output.
-t, --trace <apis>	Select which APIs to trace (ex. cuda, nvtx, osrt, cudnn, mpi)
-y, --delay <s>	Delays start of data collection by s seconds.
-d, --duration <s>	Stops data collection after s seconds.
--capture-range <trigger>	Starts data collection only after a trigger is hit (ex. nvtx).
--stats <bool>	Run nsys automatically after collection finishes.
--export <type>	Export report automatically to another format (ex. sqlite, csv, etc.)

Viewing Results

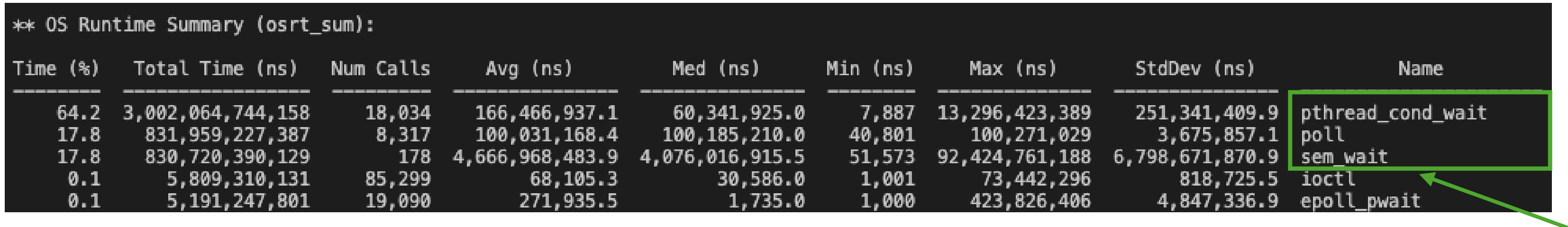
	CLI Statistics	Expert System	Local GUI*
Command	nsys stats report.nsys-rep	nsys analyze report.nsys-rep	nsys-ui report.nsys-rep
Goal	Quick summary	Interpret results	Visual performance analysis
Output	API, kernel, memory operation, push/pop range, runtime API statistics	Problems identified and suggestions to fix them	Timeline of CPU, GPU, OS activity, analysis summary, diagnostics summary, CPU samples
How	Input .nsys-rep or .sqlite via command in terminal	Input .nsys-rep or .sqlite via command in terminal	Transfer .nsys-rep to local, then run command to open with GUI manually

Analyzing Results - Summaries



```
** NVTX Range Summary (nvtx_sum):

Time (%)  Total Time (ns)  Instances      Avg (ns)           Med (ns)          Min (ns)         Max (ns)      StdDev (ns)   Style           Range
--------  ---------------  ---------  -----------------  -----------------  ---------------  ---------------  -----------  -------  ----------------------
    98.2  817,494,673,463          1  817,494,673,463.0  817,494,673,463.0  817,494,673,463  817,494,673,463          0.0  PushPop  :training
     0.9    7,802,688,212          1    7,802,688,212.0    7,802,688,212.0    7,802,688,212    7,802,688,212          0.0  PushPop  :data load/sort
     0.3    2,901,489,129          1    2,901,489,129.0    2,901,489,129.0    2,901,489,129    2,901,489,129          0.0  PushPop  :training setup
     0.3    2,142,966,883          1    2,142,966,883.0    2,142,966,883.0    2,142,966,883    2,142,966,883          0.0  PushPop  :logging data on cpu
     0.2    1,931,465,301          1    1,931,465,301.0    1,931,465,301.0    1,931,465,301    1,931,465,301          0.0  PushPop  :split/mask data
     0.0       74,112,316          1       74,112,316.0       74,112,316.0       74,112,316       74,112,316          0.0  PushPop  :initialize structures
```

Total runtime sectioned by manually defined labels

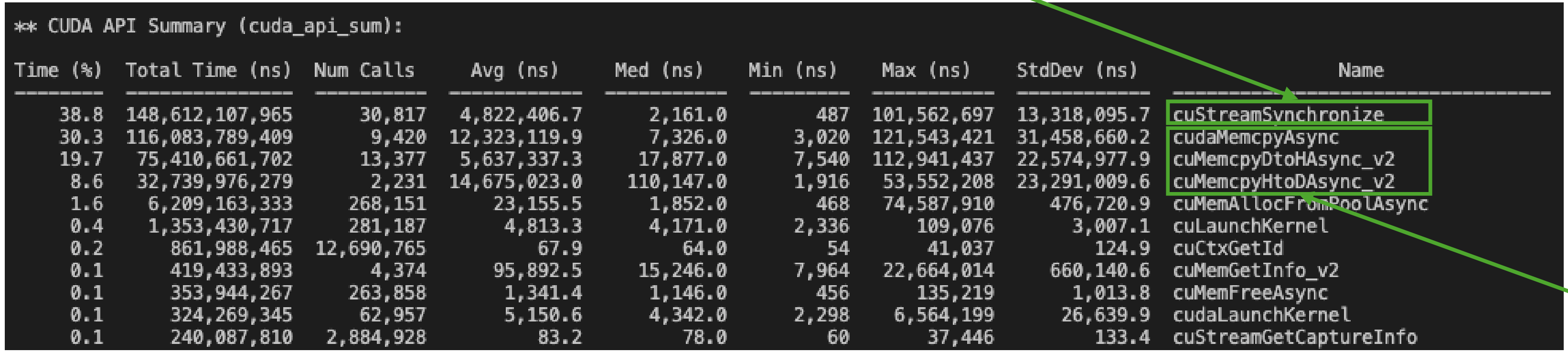


```
** OS Runtime Summary (osrt_sum):

Time (%)  Total Time (ns)  Num Calls      Avg (ns)          Med (ns)       Min (ns)     Max (ns)        StdDev (ns)          Name
--------  ---------------  ---------  ---------------  ---------------  ---------  ---------------  ---------------  ----------------
    64.2  3,002,064,744,158     18,034    166,466,937.1     60,341,925.0      7,887  13,296,423,389    251,341,409.9  pthread_cond_wait
    17.8    831,959,227,387      8,317    100,031,168.4    100,185,210.0     40,801     100,271,029      3,675,857.1  poll
    17.8    830,720,390,129        178  4,666,968,483.9  4,076,016,915.5     51,573  92,424,761,188  6,798,671,870.9  sem_wait
     0.1      5,809,310,131     85,299         68,105.3         30,586.0      1,001      73,442,296        818,725.5  ioctl
     0.1      5,191,247,801     19,090        271,935.5          1,735.0      1,000     423,826,406      4,847,336.9  epoll_pwait
```

Total runtime spent in OS calls

Wait/sleep

Wait to sync CPU with GPU



```
** CUDA API Summary (cuda_api_sum):

Time (%)  Total Time (ns)  Num Calls    Avg (ns)      Med (ns)    Min (ns)    Max (ns)     StdDev (ns)              Name
--------  ---------------  ----------  ------------  -----------  --------  -----------  ------------  ----------------------------
    38.8  148,612,107,965      30,817   4,822,406.7      2,161.0       487  101,562,697  13,318,095.7  cuStreamSynchronize
    30.3  116,083,789,409       9,420  12,323,119.9      7,326.0     3,020  121,543,421  31,458,660.2  cudaMemcpyAsync
    19.7   75,410,661,702      13,377   5,637,337.3     17,877.0     7,540  112,941,437  22,574,977.9  cuMemcpyDtoHAsync_v2
     8.6   32,739,976,279       2,231  14,675,023.0    110,147.0     1,916   53,552,208  23,291,009.6  cuMemcpyHtoDAsync_v2
     1.6    6,209,163,333     268,151      23,155.5      1,852.0       468   74,587,910     476,720.9  cuMemAllocFromPoolAsync
     0.4    1,353,430,717     281,187       4,813.3      4,171.0     2,336      109,076       3,007.1  cuLaunchKernel
     0.2      861,988,465  12,690,765          67.9         64.0        54       41,037         124.9  cuCtxGetId
     0.1      419,433,893       4,374      95,892.5     15,246.0     7,964   22,664,014     660,140.6  cuMemGetInfo_v2
     0.1      353,944,267     263,858       1,341.4      1,146.0       456      135,219       1,013.8  cuMemFreeAsync
     0.1      324,269,345      62,957       5,150.6      4,342.0     2,298    6,564,199      26,639.9  cudaLaunchKernel
     0.1      240,087,810   2,884,928          83.2         78.0        60       37,446         133.4  cuStreamGetCaptureInfo
```

Total runtime spent in CUDA API calls (CPU to GPU)

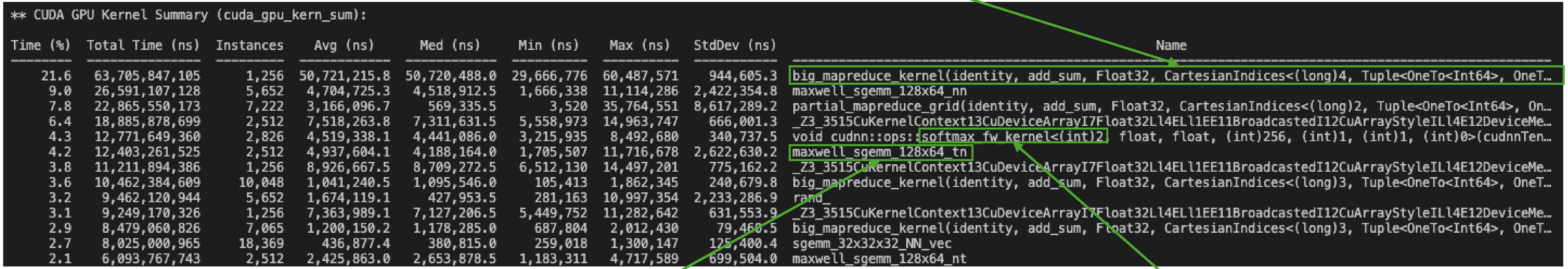
Copy between CPU and GPU

CLI Statistics (nsys stats)

Analyzing Results - Summaries

Map reduce operation

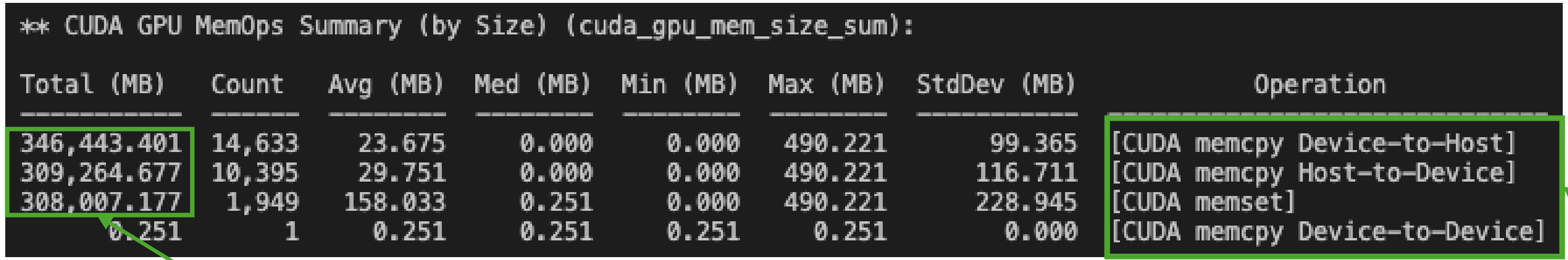
Total runtime spent on GPU kernels



```
** CUDA GPU Kernel Summary (cuda_gpu_kern_sum):

Time (%)  Total Time (ns)  Instances    Avg (ns)      Med (ns)     Min (ns)    Max (ns)    StdDev (ns)                                          Name
--------  ---------------  ---------  ------------  ------------  ----------  ----------  -------------  ----------------------------------------------------------------------------------------------------
    21.6   63,705,847,105      1,256  50,721,215.8  50,720,488.0  29,666,776  60,487,571      944,605.3  big_mapreduce_kernel(identity, add_sum, Float32, CartesianIndices<(long)4, Tuple<OneTo<Int64>, OneT…
     9.0   26,591,107,128      5,652   4,704,725.3   4,518,912.5   1,666,338  11,114,286    2,422,354.8  maxwell_sgemm_128x64_nn
     7.8   22,865,550,173      7,222   3,166,096.7     569,335.5       3,520  35,764,551    8,617,289.2  partial_mapreduce_grid(identity, add_sum, Float32, CartesianIndices<(long)2, Tuple<OneTo<Int64>, On…
     6.4   18,885,878,699      2,512   7,518,263.8   7,311,631.5   5,558,973  14,963,747      666,001.3  _Z3_3515CuKernelContext13CuDeviceArrayI7Float32Ll4ELl1EE11BroadcastedI12CuArrayStyleILl4E12DeviceMe…
     4.3   12,771,649,360      2,826   4,519,338.1   4,441,086.0   3,215,935   8,492,680      340,737.5  void cudnn::ops::softmax_fw_kernel<(int)2, float, float, (int)256, (int)1, (int)1, (int)0>(cudnnTen…
     4.2   12,403,261,525      2,512   4,937,604.1   4,188,164.0   1,705,507  11,716,678    2,622,630.2  maxwell_sgemm_128x64_tn
     3.8   11,211,894,386      1,256   8,926,667.5   8,709,272.5   6,512,130  14,497,201      775,162.2  _Z3_3515CuKernelContext13CuDeviceArrayI7Float32Ll4ELl1EE11BroadcastedI12CuArrayStyleILl4E12DeviceMe…
     3.6   10,462,384,609     10,048   1,041,240.5   1,095,546.0     105,413   1,862,345      240,679.8  big_mapreduce_kernel(identity, add_sum, Float32, CartesianIndices<(long)3, Tuple<OneTo<Int64>, OneT…
     3.2    9,462,120,944      5,652   1,674,119.1     427,953.5     281,163  10,997,354    2,233,286.9  rand_
     3.1    9,249,170,326      1,256   7,363,989.1   7,127,206.5   5,449,752  11,282,642      631,553.9  _Z3_3515CuKernelContext13CuDeviceArrayI7Float32Ll4ELl1EE11BroadcastedI12CuArrayStyleILl4E12DeviceMe…
     2.9    8,479,060,826      7,065   1,200,150.2   1,178,285.0     687,804   2,012,430       79,460.5  big_mapreduce_kernel(identity, add_sum, Float32, CartesianIndices<(long)3, Tuple<OneTo<Int64>, OneT…
     2.7    8,025,000,965     18,369     436,877.4     380,815.0     259,018   1,300,147      125,400.4  sgemm_32x32x32_NN_vec
     2.1    6,093,767,743      2,512   2,425,863.0   2,653,878.5   1,183,311   4,717,589      699,504.0  maxwell_sgemm_128x64_nt
```

Matrix multiplication

Softmax function!

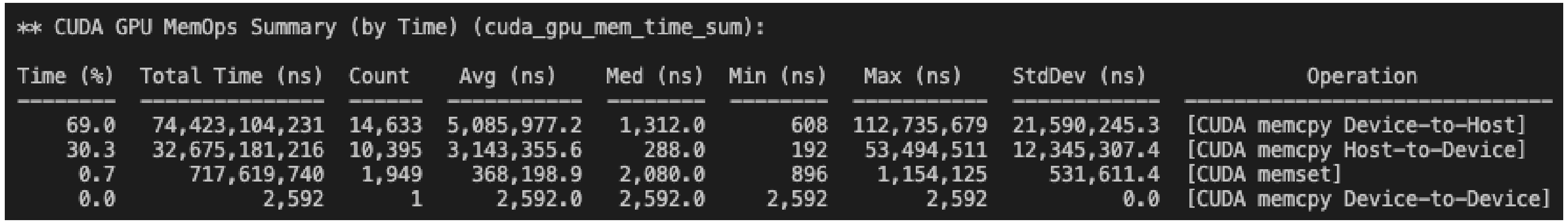


```
** CUDA GPU MemOps Summary (by Size) (cuda_gpu_mem_size_sum):

 Total (MB)  Count   Avg (MB)  Med (MB)  Min (MB)  Max (MB)  StdDev (MB)            Operation
 ----------  ------  --------  --------  --------  --------  -----------  ----------------------------
346,443.401  14,633    23.675     0.000     0.000   490.221       99.365  [CUDA memcpy Device-to-Host]
309,264.677  10,395    29.751     0.000     0.000   490.221      116.711  [CUDA memcpy Host-to-Device]
308,007.177   1,949   158.033     0.251     0.000   490.221      228.945  [CUDA memset]
      0.251       1     0.251     0.251     0.251     0.251        0.000  [CUDA memcpy Device-to-Device]
```

Total size of data copied between CPU and GPU

300GB moved!

GPU to CPU = device to host
CPU to GPU = host to device



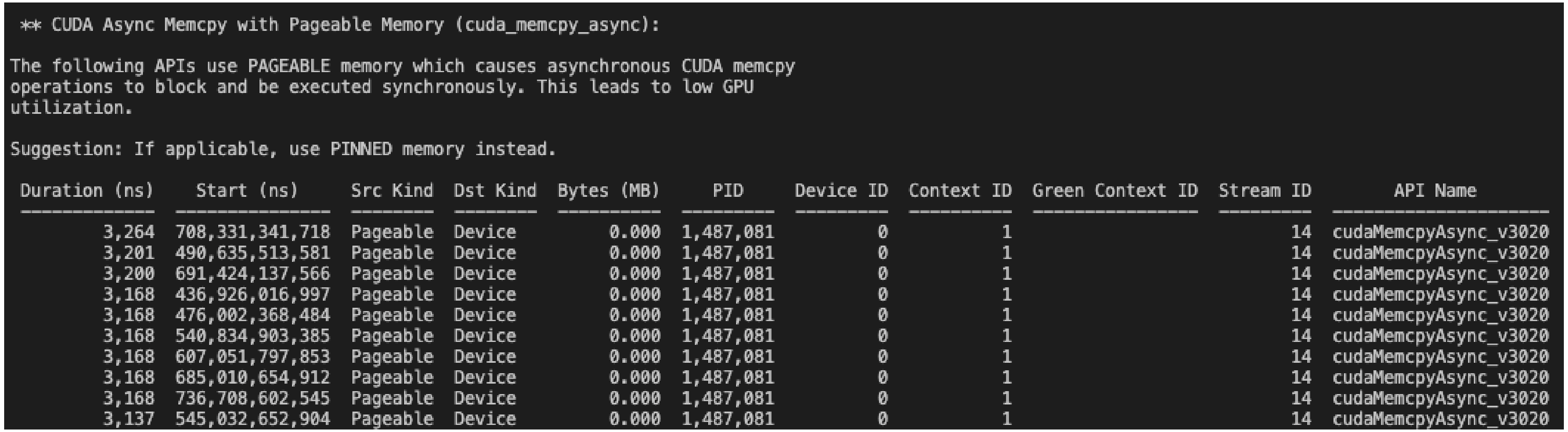
```
** CUDA GPU MemOps Summary (by Time) (cuda_gpu_mem_time_sum):

Time (%)  Total Time (ns)  Count    Avg (ns)    Med (ns)  Min (ns)   Max (ns)    StdDev (ns)            Operation
--------  ---------------  ------  -----------  --------  --------  -----------  ------------  ----------------------------
    69.0   74,423,104,231  14,633  5,085,977.2   1,312.0       608  112,735,679  21,590,245.3  [CUDA memcpy Device-to-Host]
    30.3   32,675,181,216  10,395  3,143,355.6     288.0       192   53,494,511  12,345,307.4  [CUDA memcpy Host-to-Device]
     0.7      717,619,740   1,949    368,198.9   2,080.0       896    1,154,125     531,611.4  [CUDA memset]
     0.0            2,592       1      2,592.0   2,592.0     2,592        2,592           0.0  [CUDA memcpy Device-to-Device]
```

Total runtime spent copying data between CPU and GPU

Expert System (nsys analyze)

Analyzing Results - Flags

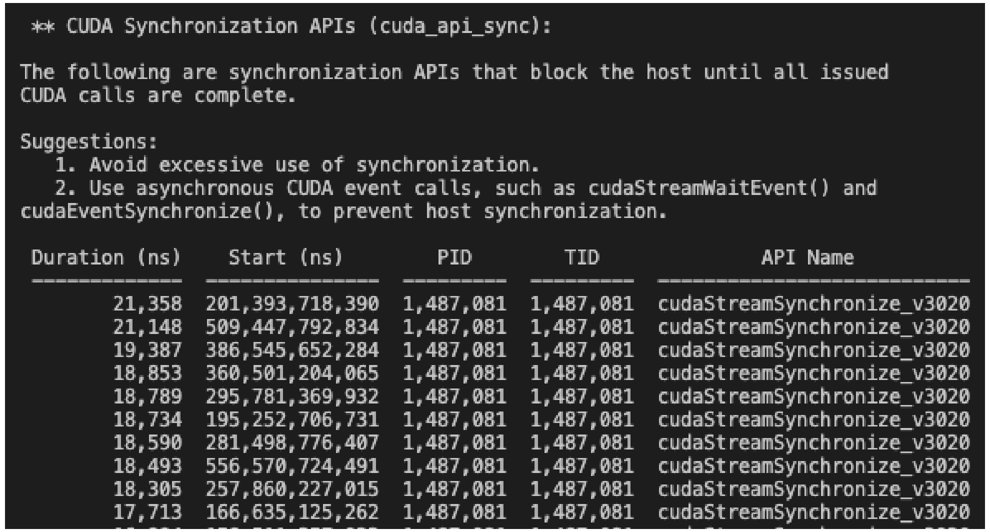
Need to align the ”Start (ns)” timestamp with GUI results and NVTX ranges to actually see/fix where these are occurring.



```
 ** CUDA Async Memcpy with Pageable Memory (cuda_memcpy_async):

The following APIs use PAGEABLE memory which causes asynchronous CUDA memcpy
operations to block and be executed synchronously. This leads to low GPU
utilization.

Suggestion: If applicable, use PINNED memory instead.

 Duration (ns)    Start (ns)      Src Kind  Dst Kind  Bytes (MB)     PID     Device ID  Context ID  Green Context ID  Stream ID        API Name
 -------------  ---------------  --------  --------  ----------  ---------  ---------  ----------  ----------------  ---------  ---------------------
         3,264  708,331,341,718  Pageable  Device         0.000  1,487,081          0           1                           14  cudaMemcpyAsync_v3020
         3,201  490,635,513,581  Pageable  Device         0.000  1,487,081          0           1                           14  cudaMemcpyAsync_v3020
         3,200  691,424,137,566  Pageable  Device         0.000  1,487,081          0           1                           14  cudaMemcpyAsync_v3020
         3,168  436,926,016,997  Pageable  Device         0.000  1,487,081          0           1                           14  cudaMemcpyAsync_v3020
         3,168  476,002,368,484  Pageable  Device         0.000  1,487,081          0           1                           14  cudaMemcpyAsync_v3020
         3,168  540,834,903,385  Pageable  Device         0.000  1,487,081          0           1                           14  cudaMemcpyAsync_v3020
         3,168  607,051,797,853  Pageable  Device         0.000  1,487,081          0           1                           14  cudaMemcpyAsync_v3020
         3,168  685,010,654,912  Pageable  Device         0.000  1,487,081          0           1                           14  cudaMemcpyAsync_v3020
         3,168  736,708,602,545  Pageable  Device         0.000  1,487,081          0           1                           14  cudaMemcpyAsync_v3020
         3,137  545,032,652,904  Pageable  Device         0.000  1,487,081          0           1                           14  cudaMemcpyAsync_v3020
```

Flags use of async copy calls from “pageable” memory

Flags where CPU code stops and waits for the GPU

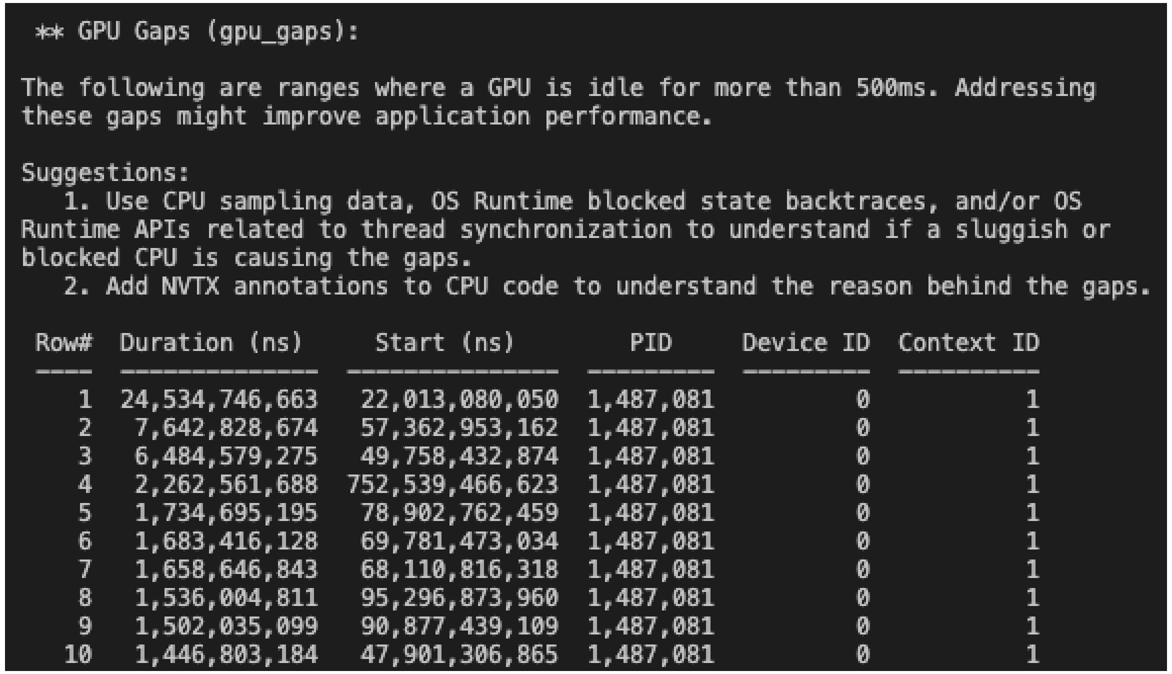


```
 ** CUDA Synchronization APIs (cuda_api_sync):

The following are synchronization APIs that block the host until all issued
CUDA calls are complete.

Suggestions:
   1. Avoid excessive use of synchronization.
   2. Use asynchronous CUDA event calls, such as cudaStreamWaitEvent() and
cudaEventSynchronize(), to prevent host synchronization.

 Duration (ns)    Start (ns)        PID        TID             API Name
 -------------  ---------------  ---------  ---------  ----------------------------
        21,358  201,393,718,390  1,487,081  1,487,081  cudaStreamSynchronize_v3020
        21,148  509,447,792,834  1,487,081  1,487,081  cudaStreamSynchronize_v3020
        19,387  386,545,652,284  1,487,081  1,487,081  cudaStreamSynchronize_v3020
        18,853  360,501,204,065  1,487,081  1,487,081  cudaStreamSynchronize_v3020
        18,789  295,781,369,932  1,487,081  1,487,081  cudaStreamSynchronize_v3020
        18,734  195,252,706,731  1,487,081  1,487,081  cudaStreamSynchronize_v3020
        18,590  281,498,776,407  1,487,081  1,487,081  cudaStreamSynchronize_v3020
        18,493  556,570,724,491  1,487,081  1,487,081  cudaStreamSynchronize_v3020
        18,305  257,860,227,015  1,487,081  1,487,081  cudaStreamSynchronize_v3020
        17,713  166,635,125,262  1,487,081  1,487,081  cudaStreamSynchronize_v3020
```

Expert System (nsys analyze)

Analyzing Results - Flags

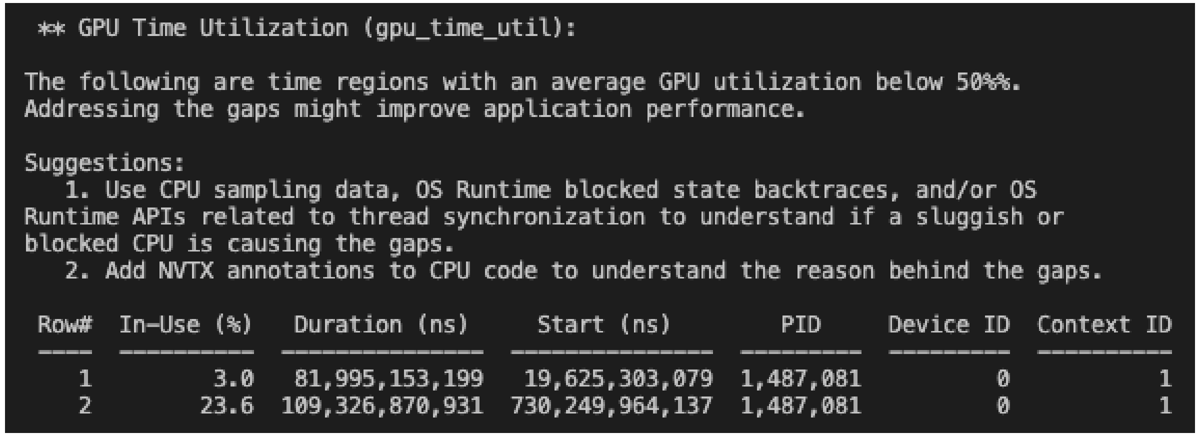
Need to align the ”Start (ns)” timestamp with GUI results and NVTX ranges to actually see/fix where these are occurring.



```
 ** GPU Gaps (gpu_gaps):

The following are ranges where a GPU is idle for more than 500ms. Addressing
these gaps might improve application performance.

Suggestions:
   1. Use CPU sampling data, OS Runtime blocked state backtraces, and/or OS
Runtime APIs related to thread synchronization to understand if a sluggish or
blocked CPU is causing the gaps.
   2. Add NVTX annotations to CPU code to understand the reason behind the gaps.

 Row#  Duration (ns)      Start (ns)        PID      Device ID  Context ID
 ----  --------------  ---------------  ---------  ---------  ----------
    1  24,534,746,663   22,013,080,050  1,487,081          0           1
    2   7,642,828,674   57,362,953,162  1,487,081          0           1
    3   6,484,579,275   49,758,432,874  1,487,081          0           1
    4   2,262,561,688  752,539,466,623  1,487,081          0           1
    5   1,734,695,195   78,902,762,459  1,487,081          0           1
    6   1,683,416,128   69,781,473,034  1,487,081          0           1
    7   1,658,646,843   68,110,816,318  1,487,081          0           1
    8   1,536,004,811   95,296,873,960  1,487,081          0           1
    9   1,502,035,099   90,877,439,109  1,487,081          0           1
   10   1,446,803,184   47,901,306,865  1,487,081          0           1
```

Flags where GPU was found to be idle for >0.5s

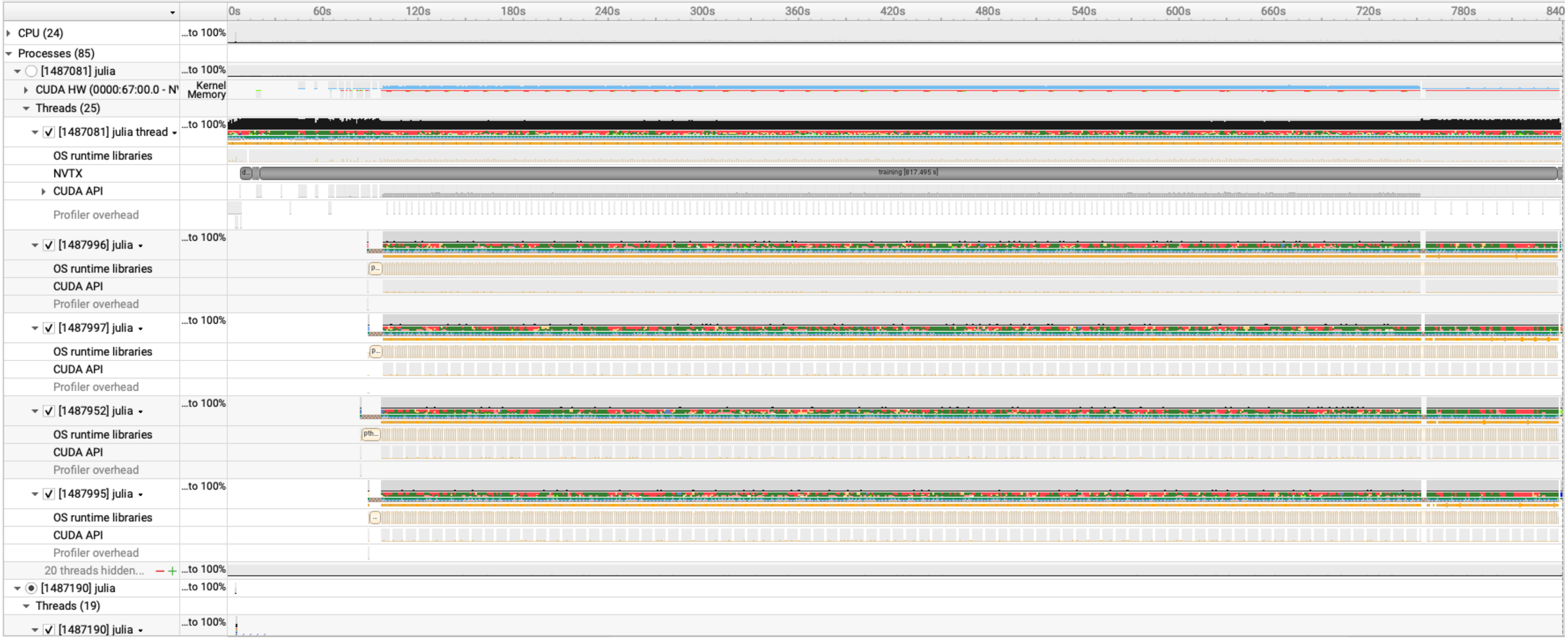
Flags where GPU was sub-optimally utilized



```
 ** GPU Time Utilization (gpu_time_util):

The following are time regions with an average GPU utilization below 50%%.
Addressing the gaps might improve application performance.

Suggestions:
   1. Use CPU sampling data, OS Runtime blocked state backtraces, and/or OS
Runtime APIs related to thread synchronization to understand if a sluggish or
blocked CPU is causing the gaps.
   2. Add NVTX annotations to CPU code to understand the reason behind the gaps.

 Row#  In-Use (%)   Duration (ns)       Start (ns)        PID      Device ID  Context ID
 ----  ----------  ---------------  ---------------  ---------  ---------  ----------
    1         3.0   81,995,153,199   19,625,303,079  1,487,081          0           1
    2        23.6  109,326,870,931  730,249,964,137  1,487,081          0           1
```

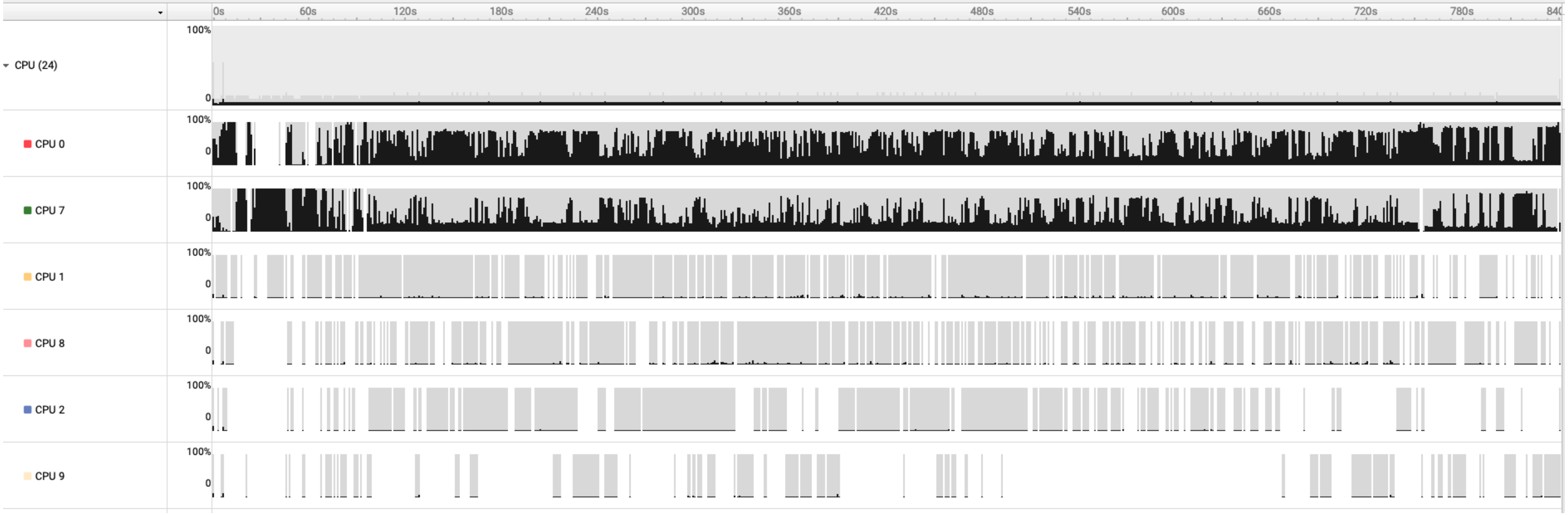
Local GUI (nsys-ui)

Analyzing Results - Timeline View



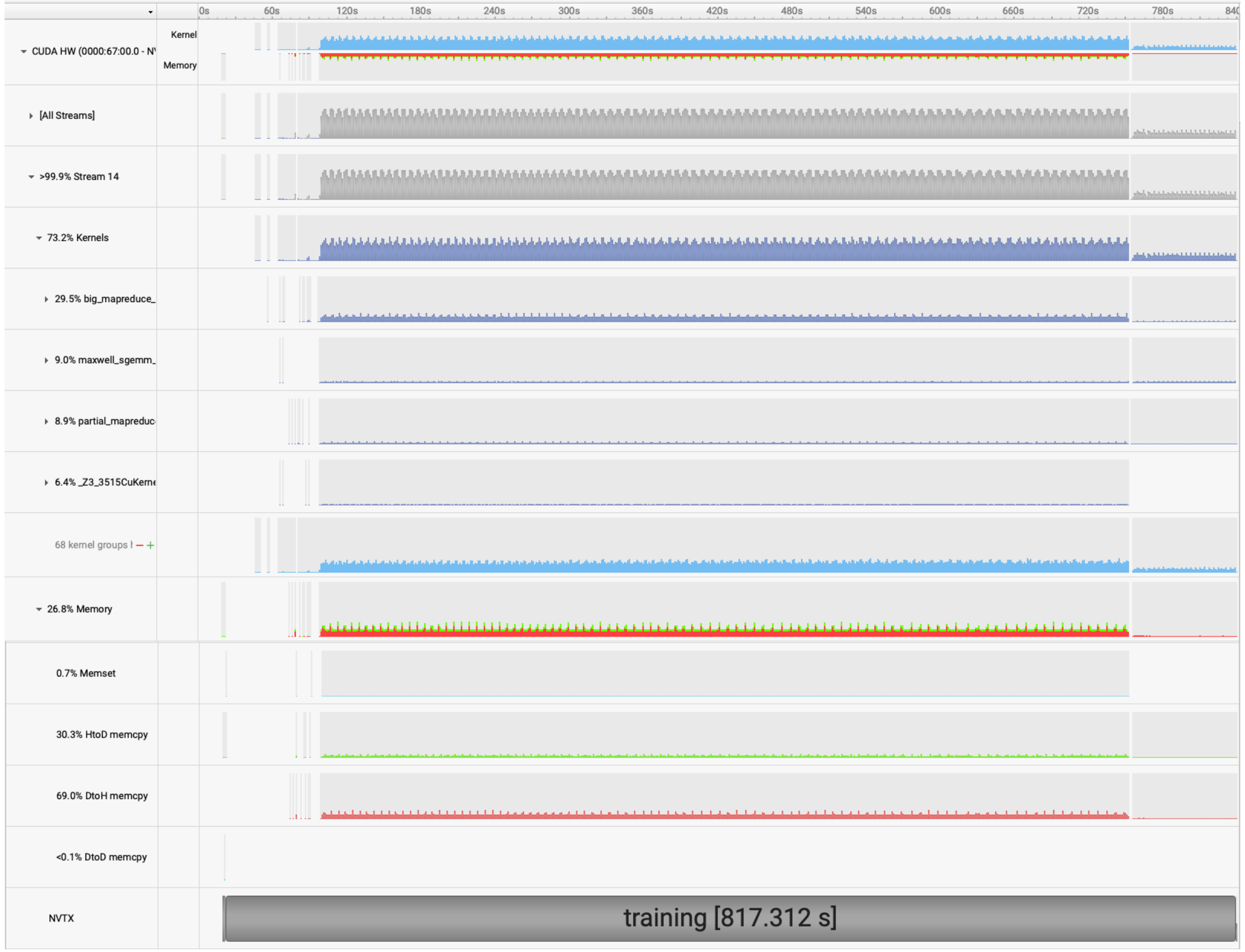
Local GUI (nsys-ui)

Analyzing Results - Timeline View (CPU)



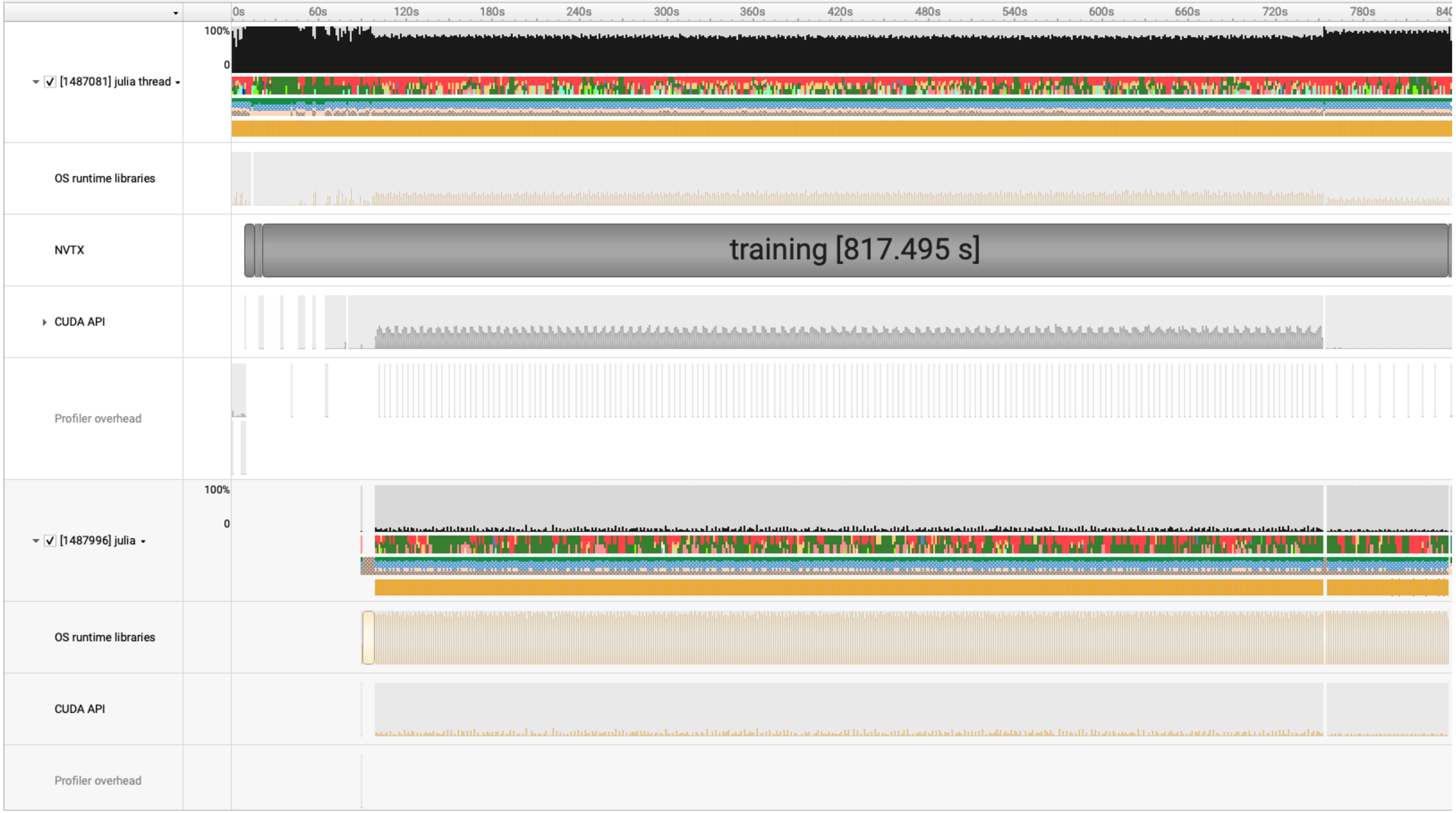
Local GUI (nsys-ui)

Analyzing Results - Timeline View (?)



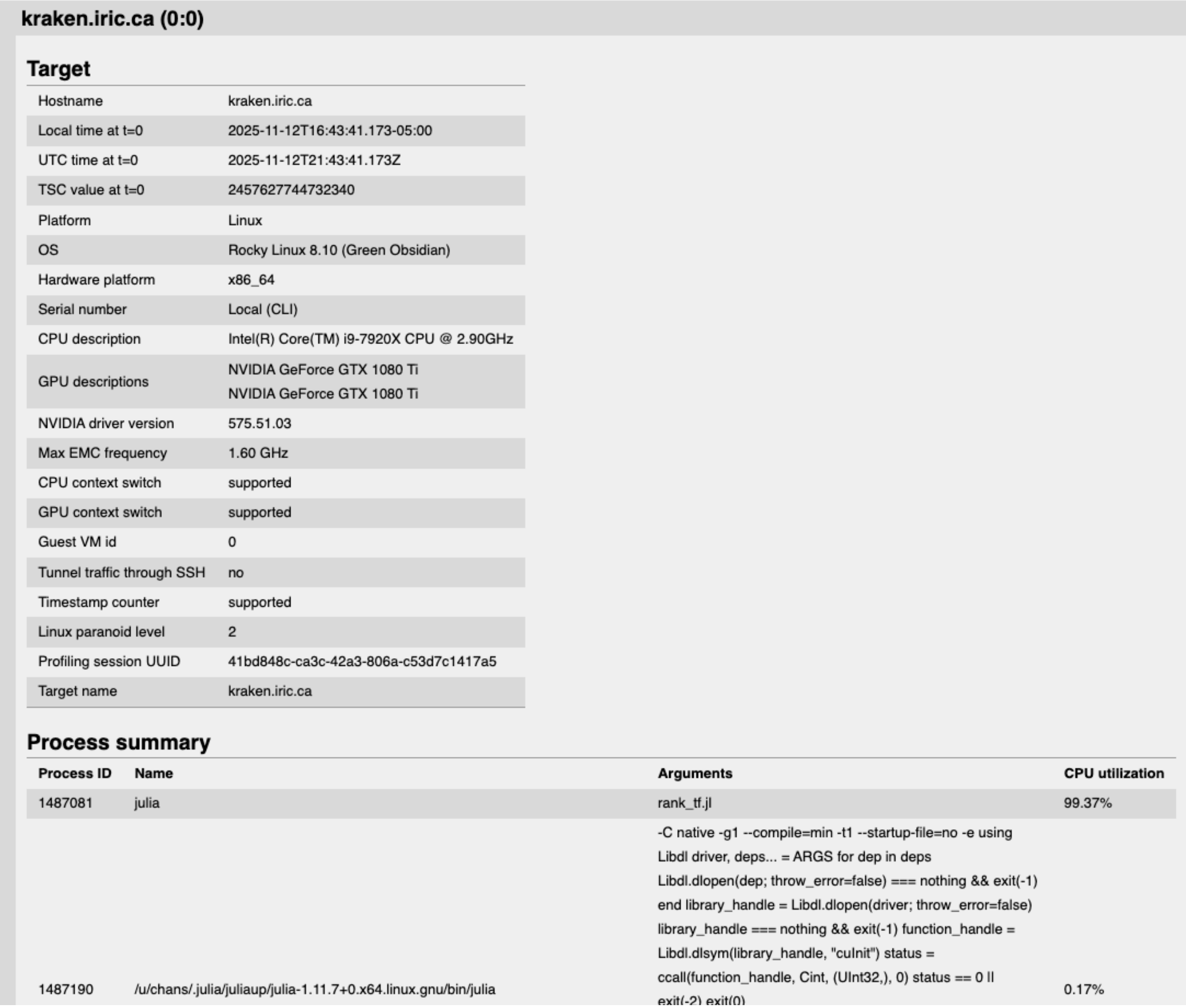
Local GUI (nsys-ui)

Analyzing Results - Timeline View (?)



Local GUI (nsys-ui)

Analyzing Results - Analysis Summary



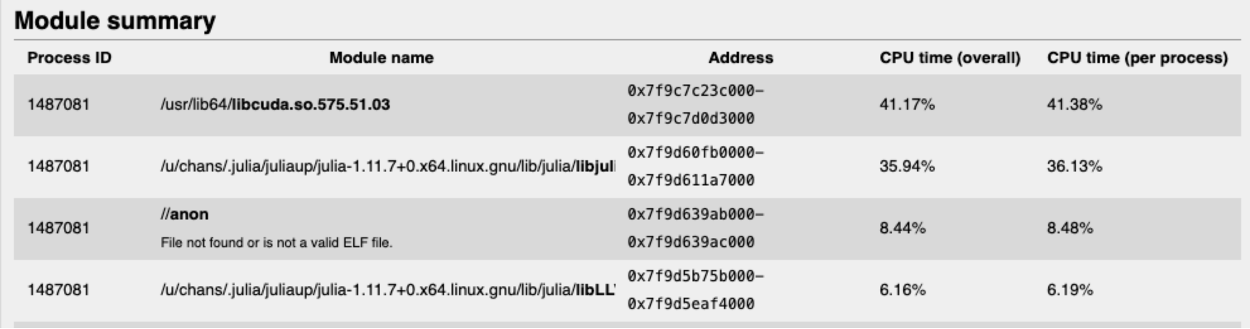
kraken.iric.ca (0:0)

Target

Hostname	kraken.iric.ca
Local time at t=0	2025-11-12T16:43:41.173-05:00
UTC time at t=0	2025-11-12T21:43:41.173Z
TSC value at t=0	2457627744732340
Platform	Linux
OS	Rocky Linux 8.10 (Green Obsidian)
Hardware platform	x86_64
Serial number	Local (CLI)
CPU description	Intel(R) Core(TM) i9-7920X CPU @ 2.90GHz
GPU descriptions	NVIDIA GeForce GTX 1080 Ti NVIDIA GeForce GTX 1080 Ti
NVIDIA driver version	575.51.03
Max EMC frequency	1.60 GHz
CPU context switch	supported
GPU context switch	supported
Guest VM id	0
Tunnel traffic through SSH	no
Timestamp counter	supported
Linux paranoid level	2
Profiling session UUID	41bd848c-ca3c-42a3-806a-c53d7c1417a5
Target name	kraken.iric.ca

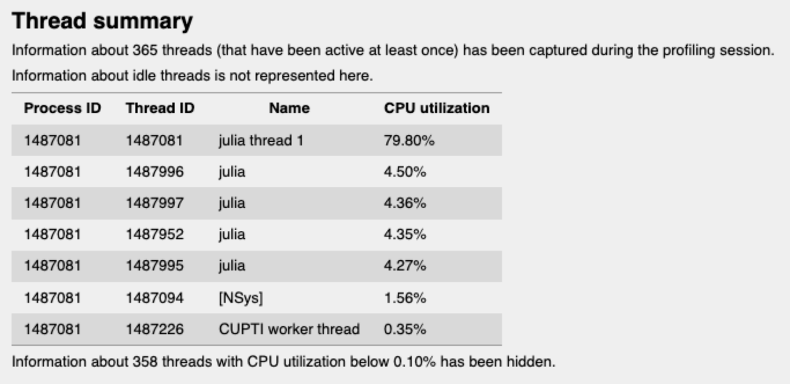
Process summary

Process ID	Name	Arguments	CPU utilization
1487081	julia	rank_tf.jl	99.37%
1487190	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/bin/julia	-C native -g1 --compile=min -t1 --startup-file=no -e using Libdl driver, deps... = ARGS for dep in deps Libdl.dlopen(dep; throw_error=false) === nothing && exit(-1) end library_handle = Libdl.dlopen(driver; throw_error=false) library_handle === nothing && exit(-1) function_handle = Libdl.dlsym(library_handle, "cuInit") status = ccall(function_handle, Cint, (UInt32,), 0) status == 0 || exit(-2) exit(0)	0.17%



Module summary

Process ID	Module name	Address	CPU time (overall)	CPU time (per process)
1487081	/usr/lib64/**libcuda.so.575.51.03**	0x7f9c7c23c000– 0x7f9c7d0d3000	41.17%	41.38%
1487081	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/lib/julia/**libjul**	0x7f9d60fb0000– 0x7f9d611a7000	35.94%	36.13%
1487081	**//anon** File not found or is not a valid ELF file.	0x7f9d639ab000– 0x7f9d639ac000	8.44%	8.48%
1487081	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/lib/julia/**libLL**	0x7f9d5b75b000– 0x7f9d5eaf4000	6.16%	6.19%



Thread summary

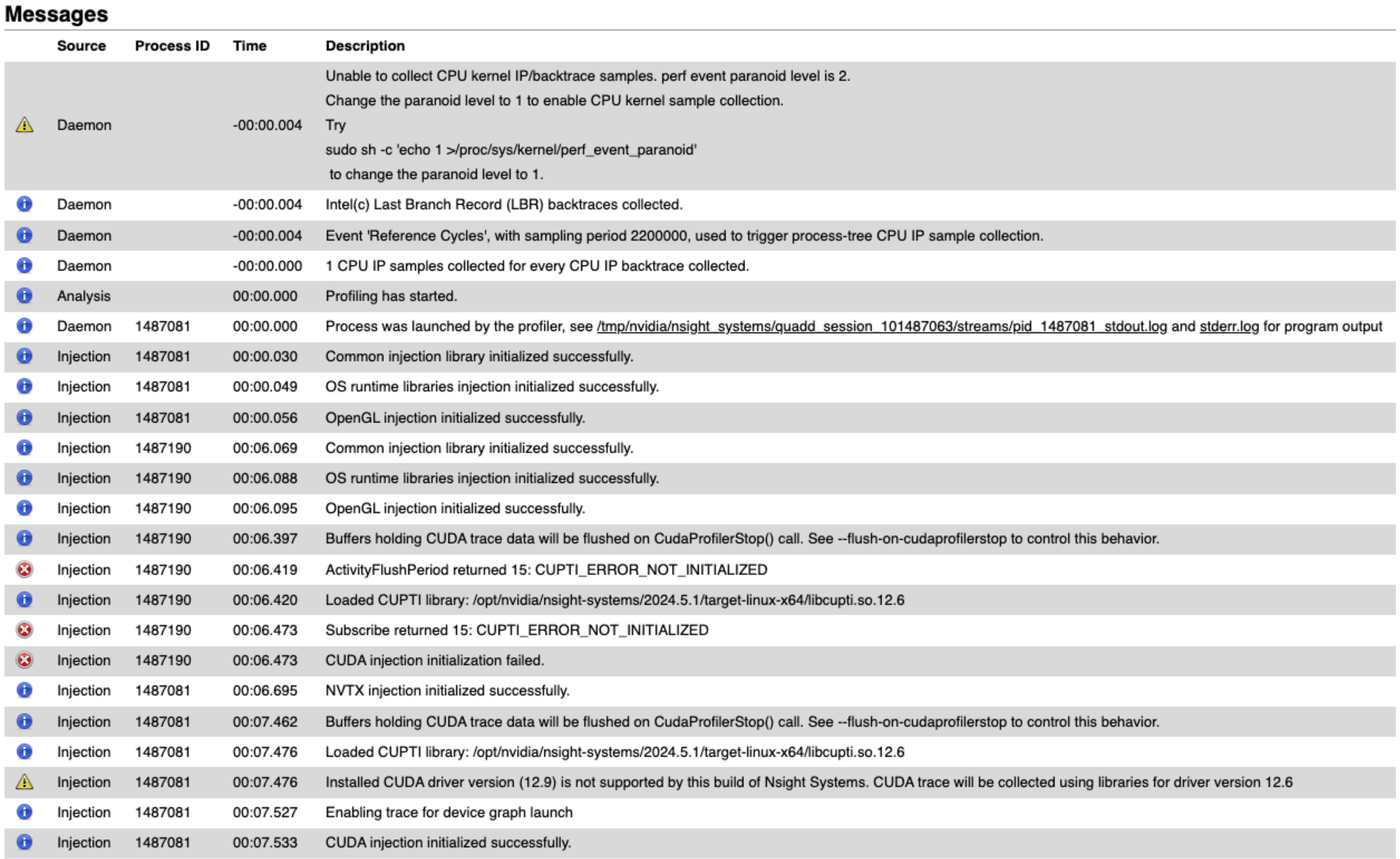
Information about 365 threads (that have been active at least once) has been captured during the profiling session. Information about idle threads is not represented here.

Process ID	Thread ID	Name	CPU utilization
1487081	1487081	julia thread 1	79.80%
1487081	1487996	julia	4.50%
1487081	1487997	julia	4.36%
1487081	1487952	julia	4.35%
1487081	1487995	julia	4.27%
1487081	1487094	[NSys]	1.56%
1487081	1487226	CUPTI worker thread	0.35%

Information about 358 threads with CPU utilization below 0.10% has been hidden.

Local GUI (nsys-ui)

Analyzing Results - Diagnostics Summary

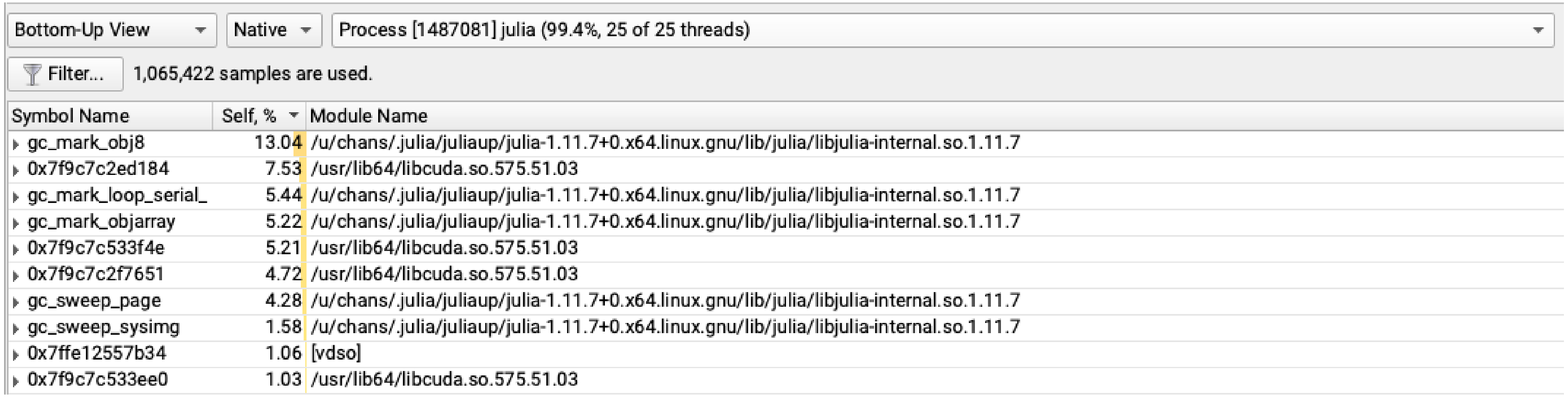


Messages

	Source	Process ID	Time	Description
⚠	Daemon		-00:00.004	Unable to collect CPU kernel IP/backtrace samples. perf event paranoid level is 2. Change the paranoid level to 1 to enable CPU kernel sample collection. Try sudo sh -c 'echo 1 >/proc/sys/kernel/perf_event_paranoid' to change the paranoid level to 1.
ℹ	Daemon		-00:00.004	Intel(c) Last Branch Record (LBR) backtraces collected.
ℹ	Daemon		-00:00.004	Event 'Reference Cycles', with sampling period 2200000, used to trigger process-tree CPU IP sample collection.
ℹ	Daemon		-00:00.000	1 CPU IP samples collected for every CPU IP backtrace collected.
ℹ	Analysis		00:00.000	Profiling has started.
ℹ	Daemon	1487081	00:00.000	Process was launched by the profiler, see /tmp/nvidia/nsight_systems/quadd_session_101487063/streams/pid_1487081_stdout.log and stderr.log for program output
ℹ	Injection	1487081	00:00.030	Common injection library initialized successfully.
ℹ	Injection	1487081	00:00.049	OS runtime libraries injection initialized successfully.
ℹ	Injection	1487081	00:00.056	OpenGL injection initialized successfully.
ℹ	Injection	1487190	00:06.069	Common injection library initialized successfully.
ℹ	Injection	1487190	00:06.088	OS runtime libraries injection initialized successfully.
ℹ	Injection	1487190	00:06.095	OpenGL injection initialized successfully.
ℹ	Injection	1487190	00:06.397	Buffers holding CUDA trace data will be flushed on CudaProfilerStop() call. See --flush-on-cudaprofilerstop to control this behavior.
⊗	Injection	1487190	00:06.419	ActivityFlushPeriod returned 15: CUPTI_ERROR_NOT_INITIALIZED
ℹ	Injection	1487190	00:06.420	Loaded CUPTI library: /opt/nvidia/nsight-systems/2024.5.1/target-linux-x64/libcupti.so.12.6
⊗	Injection	1487190	00:06.473	Subscribe returned 15: CUPTI_ERROR_NOT_INITIALIZED
⊗	Injection	1487190	00:06.473	CUDA injection initialization failed.
ℹ	Injection	1487081	00:06.695	NVTX injection initialized successfully.
ℹ	Injection	1487081	00:07.462	Buffers holding CUDA trace data will be flushed on CudaProfilerStop() call. See --flush-on-cudaprofilerstop to control this behavior.
ℹ	Injection	1487081	00:07.476	Loaded CUPTI library: /opt/nvidia/nsight-systems/2024.5.1/target-linux-x64/libcupti.so.12.6
⚠	Injection	1487081	00:07.476	Installed CUDA driver version (12.9) is not supported by this build of Nsight Systems. CUDA trace will be collected using libraries for driver version 12.6
ℹ	Injection	1487081	00:07.527	Enabling trace for device graph launch
ℹ	Injection	1487081	00:07.533	CUDA injection initialized successfully.

Local GUI (nsys-ui)

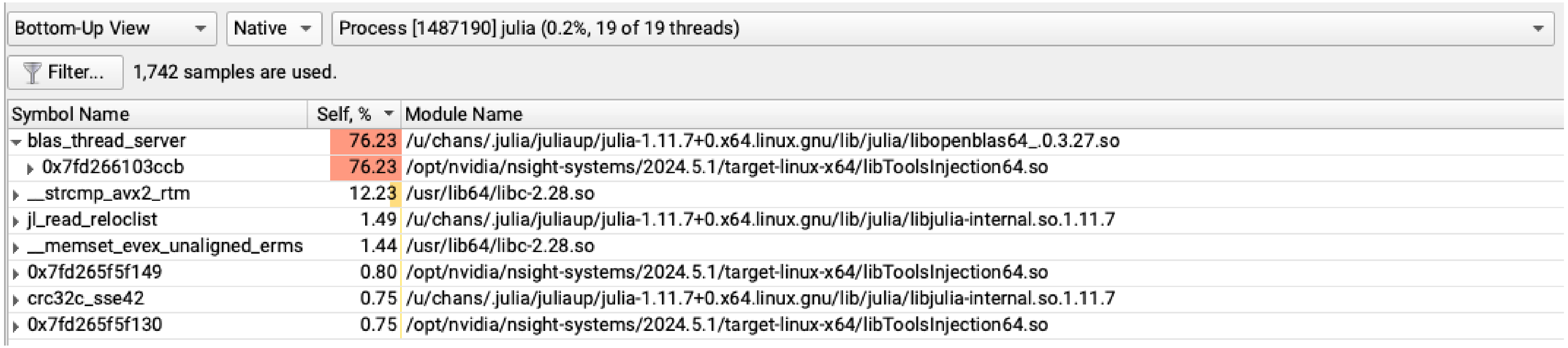
Analyzing Results - ?



Bottom-Up View | Native | Process [1487081] julia (99.4%, 25 of 25 threads)

1,065,422 samples are used.

Symbol Name	Self, %	Module Name
gc_mark_obj8	13.04	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/lib/julia/libjulia-internal.so.1.11.7
0x7f9c7c2ed184	7.53	/usr/lib64/libcuda.so.575.51.03
gc_mark_loop_serial_	5.44	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/lib/julia/libjulia-internal.so.1.11.7
gc_mark_objarray	5.22	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/lib/julia/libjulia-internal.so.1.11.7
0x7f9c7c533f4e	5.21	/usr/lib64/libcuda.so.575.51.03
0x7f9c7c2f7651	4.72	/usr/lib64/libcuda.so.575.51.03
gc_sweep_page	4.28	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/lib/julia/libjulia-internal.so.1.11.7
gc_sweep_sysimg	1.58	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/lib/julia/libjulia-internal.so.1.11.7
0x7ffe12557b34	1.06	[vdso]
0x7f9c7c533ee0	1.03	/usr/lib64/libcuda.so.575.51.03



Bottom-Up View | Native | Process [1487190] julia (0.2%, 19 of 19 threads)

1,742 samples are used.

Symbol Name	Self, %	Module Name
blas_thread_server	76.23	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/lib/julia/libopenblas64_.0.3.27.so
0x7fd266103ccb	76.23	/opt/nvidia/nsight-systems/2024.5.1/target-linux-x64/libToolsInjection64.so
__strcmp_avx2_rtm	12.23	/usr/lib64/libc-2.28.so
jl_read_reloclist	1.49	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/lib/julia/libjulia-internal.so.1.11.7
__memset_evex_unaligned_erms	1.44	/usr/lib64/libc-2.28.so
0x7fd265f5f149	0.80	/opt/nvidia/nsight-systems/2024.5.1/target-linux-x64/libToolsInjection64.so
crc32c_sse42	0.75	/u/chans/.julia/juliaup/julia-1.11.7+0.x64.linux.gnu/lib/julia/libjulia-internal.so.1.11.7
0x7fd265f5f130	0.75	/opt/nvidia/nsight-systems/2024.5.1/target-linux-x64/libToolsInjection64.so

Demo :D