

# GPU Profiling on Expanse

## March 12, 2021

Mahidhar Tatineni  
SDSC

EXPANSE  
COMPUTING WITHOUT BOUNDARIES

SAN DIEGO SUPERCOMPUTER CENTER



NSF Award 1928224

# Outline

- **nvprof**
  - Overview of options
  - Use CUDA SDK examples to illustrate profiling process
  - Import/export of profiles
  - Openacc example
- **Nsight Systems**
  - CLI and GUI options
  - Use CUDA SDK and vector addition examples to illustrate offline GUI use
- **Nsight Compute** – brief overview; currently disabled on Expanse

# NVIDIA nvprof

- Command line tool that profiles CUDA related activities on both CPU and GPU.
- Console display of results OR can be saved for viewing using nvprof or nvvp (Visual Profiler).
- **CUDA options:** related to events, metrics, kernels, memory, source level analysis, and dependency analysis.
- **CPU Options:** related to frequency, depth, mode, scope, openacc, and openmp.
- **Print and IO Options:** related to demangling, summary, import/export of profiles, log files.
- Detailed options in nvprof documentation:
  - <https://docs.nvidia.com/cuda/profiler-users-guide/index.html#nvprof-overview>

# NVIDIA nvprof: Query events, metrics

```
[mahidhar@exp-14-59 ~]$ nvprof --query-metrics
```

## Available Metrics:

Name	Description
------	-------------

Device 0 (Tesla V100-SXM2-32GB):

**inst\_per\_warp:** Average number of instructions executed by each warp

**branch\_efficiency:** Ratio of branch instruction to sum of branch and divergent branch instruction

**warp\_execution\_efficiency:** Ratio of the average active threads per warp to the maximum number of threads per warp supported on a multiprocessor

**warp\_nonpred\_execution\_efficiency:** Ratio of the average active threads per warp executing non-predicated instructions to the maximum number of threads per warp supported on a multiprocessor

**inst\_replay\_overhead:** Average number of replays for each instruction executed

...

...

```
[mahidhar@exp-14-59 ~]$ nvprof --query-events | more
```

## Available Events:

Name	Description
------	-------------

Device 0 (Tesla V100-SXM2-32GB):

Domain domain\_a:

**active\_cycles\_pm:** Number of cycles a multiprocessor has at least one active warp.

**active\_warps\_pm:** Accumulated number of active warps per cycle. For every cycle it increments by the number of active warps in the cycle which can be in the range 0 to 64.

**shared\_Id\_transactions:** Number of transactions for shared load accesses. Maximum transaction size in volta is

128 bytes, any warp accessing more than 128 bytes will cause multiple transactions for a shared load instruction. This also includes extra transactions caused by shared bank conflicts.

**shared\_st\_transactions:** Number of transactions for shared store accesses. Maximum transaction size in volta is 128 bytes, any warp accessing more than 128 bytes will cause multiple transactions for a shared store instruction. This also includes extra transactions caused by shared bank conflicts.

...

...

# NVIDIA nvprof: Simple Example

```
[mahidhar@exp-14-59 matrixMul]$ nvprof ./matrixMul
[Matrix Multiply Using CUDA] - Starting...
==29356== NVPROF is profiling process 29356, command: ./matrixMul

GPU Device 0: "Volta" with compute capability 7.0

MatrixA(320,320), MatrixB(640,320)
Computing result using CUDA Kernel...
done
Performance= 2764.50 GFlop/s, Time= 0.047 msec, Size= 131072000 Ops, WorkgroupSize=
1024 threads/block
Checking computed result for correctness: Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary
when GPU Boost is enabled.
==29356== Profiling application: ./matrixMul
==29356== Profiling result:
      Type Time(%)   Time   Calls    Avg     Min     Max   Name
GPU activities: 98.80% 13.986ms    301 46.465us 46.175us 47.263us void
MatrixMulCUDA<int=32>(float*, float*, float*, int, int)
      0.75% 105.98us    2 52.991us 36.831us 69.151us [CUDA memcpy HtoD]
      0.45% 63.392us    1 63.392us 63.392us 63.392us [CUDA memcpy DtoH]
API calls: 89.03% 175.52ms    3 58.507ms 2.3030us 175.52ms cudaMalloc
      6.55% 12.911ms    1 12.911ms 12.911ms 12.911ms cudaEventSynchronize
      1.99% 3.9151ms    4 978.78us 857.05us 1.0630ms cuDeviceTotalMem
      0.70% 1.3831ms   301 4.5950us 3.4940us 170.29us cudaLaunchKernel
```

0.50%	988.44us	388	2.5470us	112ns	116.15us	cuDeviceGetAttribute
0.49%	974.78us	3	324.93us	172.52us	552.90us	cudaMemcpyAsync
0.22%	431.51us	22	19.614us	311ns	115.63us	cudaDeviceGetAttribute
0.17%	340.11us	1	340.11us	340.11us	340.11us	cudaGetDeviceCount
0.10%	199.36us	1	199.36us	199.36us	199.36us	cudaSetDevice
0.07%	145.48us	3	48.492us	3.0960us	130.70us	cudaFree
0.07%	143.04us	2	71.520us	612ns	142.43us	cudaEventCreate
0.05%	104.37us	4	26.091us	21.334us	37.046us	cuDeviceGetName
0.02%	48.915us	2	24.457us	3.0980us	45.817us	cudaStreamSynchronize
0.01%	12.254us	4	3.0630us	2.0060us	4.9760us	cuDeviceGetPCIBusId
0.00%	8.5450us	1	8.5450us	8.5450us	8.5450us	cudaStreamCreateWithFlags
0.00%	6.4310us	3	2.1430us	465ns	4.2790us	cuDeviceGetCount
0.00%	6.3390us	2	3.1690us	2.2410us	4.0980us	cudaEventRecord
0.00%	2.5880us	1	2.5880us	2.5880us	2.5880us	cudaEventElapsedTime
0.00%	2.1770us	2	1.0880us	435ns	1.7420us	cudaEventDestroy
0.00%	1.6600us	8	207ns	106ns	759ns	cuDeviceGet
0.00%	682ns	4	170ns	143ns	227ns	cuDeviceGetUuid

# NVIDIA nvprof – Examples on Expanse

- Copy CUDA examples:

```
cp -r /cm/shared/apps/cuda10.2/sdk/10.2.89 $HOME/examples
```

- Setup profiling environment:

```
module load cuda10.2/toolkit
```

```
module load cuda10.2/profiler
```

- Compile some cases:

```
cd $HOME/examples/0_Simple/matrixMul
```

```
make
```

```
cd $HOME/examples/7_CUDALibraries/cuSolverSp_LinearSolver
```

```
make
```

# NVIDIA nvprof

- Get interactive access to a GPU node for 15 minutes:

```
srun --pty --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 -p gpu-shared --gpus=1 -A XYZ123 -t 00:15:00 --wait 0 /bin/bash
```

```
module reset; module load cuda10.2/toolkit cuda10.2/profiler
```

- Get timeline of activities on GPU in chronological order:

```
nvprof --print-gpu-trace ./cuSolverSp_LinearSolver
```

- Get timeline of all CUDA runtime, driver API calls invoked on the host in chronological order:

```
nvprof --print-api-trace ./cuSolverSp_LinearSolver
```

- Collect info on selected events, metrics:

```
nvprof --events warps_launched,active_warps_pm,inst_issued0 --metrics flop_count_dp ./cuSolverSp_LinearSolver
```

*(Last one needs permissions to be set up – we are working on this)*

# NVIDIA nvprof

- System profiling: Low frequency sampling of the power, clock, and thermal behavior of GPU

**nvprof --print-gpu-trace --system-profiling on ./cuSolverSp\_LinearSolver**

- Dependency analysis:

**nvprof --dependency-analysis ./cuSolverSp\_LinearSolver**

- OpenACC profiling: (copy over /cm/shared/examples/sdsc/openacc)

**nvprof ./laplace2d.openacc.exe**

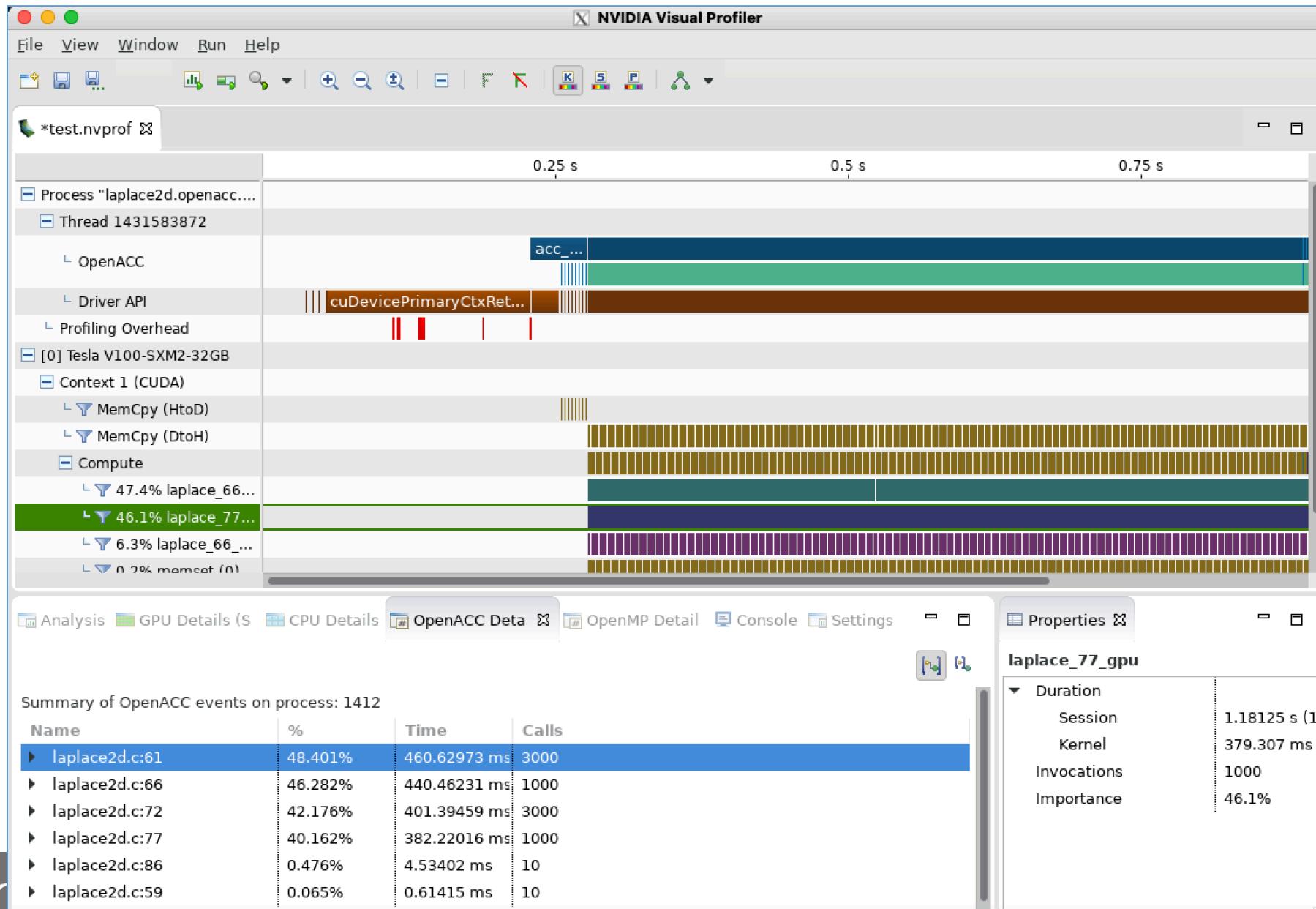
**nvprof --print-openacc-summary ./laplace2d.openacc.exe**

**nvprof --print-openacc-trace ./laplace2d.openacc.exe**

- Export to file that can be opened with nvvp

**nvprof --export-profile test.nvprof --print-openacc-trace ./laplace2d.openacc.exe**

# Reading nvprof result from nvvp



# Nsight Systems

- Systemwide performance tool, profile combines info from both CPU and GPU.
- Gives a full system view of the code execution and helps identify bottlenecks at all levels. Examples:
  - GPU starvation
  - unnecessary GPU synchronization
  - insufficient CPU parallelizing
  - memory access/data movement inefficiencies
- Can run in both command line (CLI) and graphical (GUI) mode.
- CLI based runs can export report files that can viewed on client machines.

# Nsight Systems

- Several tracing options. Examples: cublas, cuda, cudnn, nvtx, opengl, openacc, openmp, osrt, and mpi.
- OpenMPI and MPICH options.
- CUDA and OS runtime backtrace options.
- Sample screenshots available at:

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

- **Currently on Expanse there are some limitations in place due to security reasons.**
- **Recommend downloading a copy on your desktop/laptop (need developer registration with NVIDIA) if you want to use GUI.**

# Nsight Systems: Simple Profiling Example

- Get interactive access on a GPU node:

```
srun --pty --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 -p gpu-shared --  
gpus=1 -A XYZ123 -t 00:15:00 --wait 0 /bin/bash
```

- Set up profiling environment:

```
module reset
```

```
module load cuda10.2/toolkit
```

```
export PATH=/cm/shared/examples/sdsc/nsight/v2020.5/bin:$PATH
```

- Profile with cli and write to baseline report file:

```
nsys profile -t cuda,osrt -o baseline -w true ./cuSolverSp_LinearSolver
```

# Nsight Systems: Simple Profiling Example

```
[mahidhar_test@exp-1-57 cuSolverSp_LinearSolver]$ nsys profile -t cuda,osrt -o baseline  
-w true ./cuSolverSp_LinearSolver
```

Collecting data...

GPU Device 0: "Volta" with compute capability 7.0

Using default input file [./lap2D\_5pt\_n100.mtx]

step 1: read matrix market format

sparse matrix A is 10000 x 10000 with 49600 nonzeros, base=1

step 2: reorder the matrix A to minimize zero fill-in

    if the user choose a reordering by -P=symrcm, -P=symamd or -P=metis

step 2.1: no reordering is chosen, Q = 0:n-1

step 2.2: B = A(Q,Q)

step 3: b(j) = 1 + j/n

step 4: prepare data on device

step 5: solve A\*x = b on CPU

step 6: evaluate residual r = b - A\*x (result on CPU)

(CPU) |b - A\*x| = 5.456968E-12

(CPU) |A| = 8.000000E+00

(CPU) |x| = 1.136492E+03

(CPU) |b| = 1.999900E+00

(CPU) |b - A\*x|/(|A|\*|x| + |b|) = 6.000667E-16

step 7: solve A\*x = b on GPU

step 8: evaluate residual r = b - A\*x (result on GPU)

(GPU) |b - A\*x| = 1.818989E-12

(GPU) |A| = 8.000000E+00

(GPU) |x| = 1.136492E+03

(GPU) |b| = 1.999900E+00

(GPU) |b - A\*x|/(|A|\*|x| + |b|) = 2.000222E-16

timing chol: CPU = 0.162645 sec , GPU = 0.086416 sec

show last 10 elements of solution vector (GPU)

consistent result for different reordering and solver

x[9990] = 3.000016E+01

x[9991] = 2.807343E+01

x[9992] = 2.601354E+01

x[9993] = 2.380285E+01

x[9994] = 2.141866E+01

x[9995] = 1.883070E+01

x[9996] = 1.599668E+01

x[9997] = 1.285365E+01

x[9998] = 9.299423E+00

x[9999] = 5.147265E+00

Processing events...

Saving temporary "/tmp/nsys-report-940c-3a9e-7049-6092.qdstrm" file to disk...

Creating final output files...

Processing [=====100%]

Saved report file to "/tmp/nsys-report-940c-3a9e-7049-6092.qdrep"

Unable to create output file

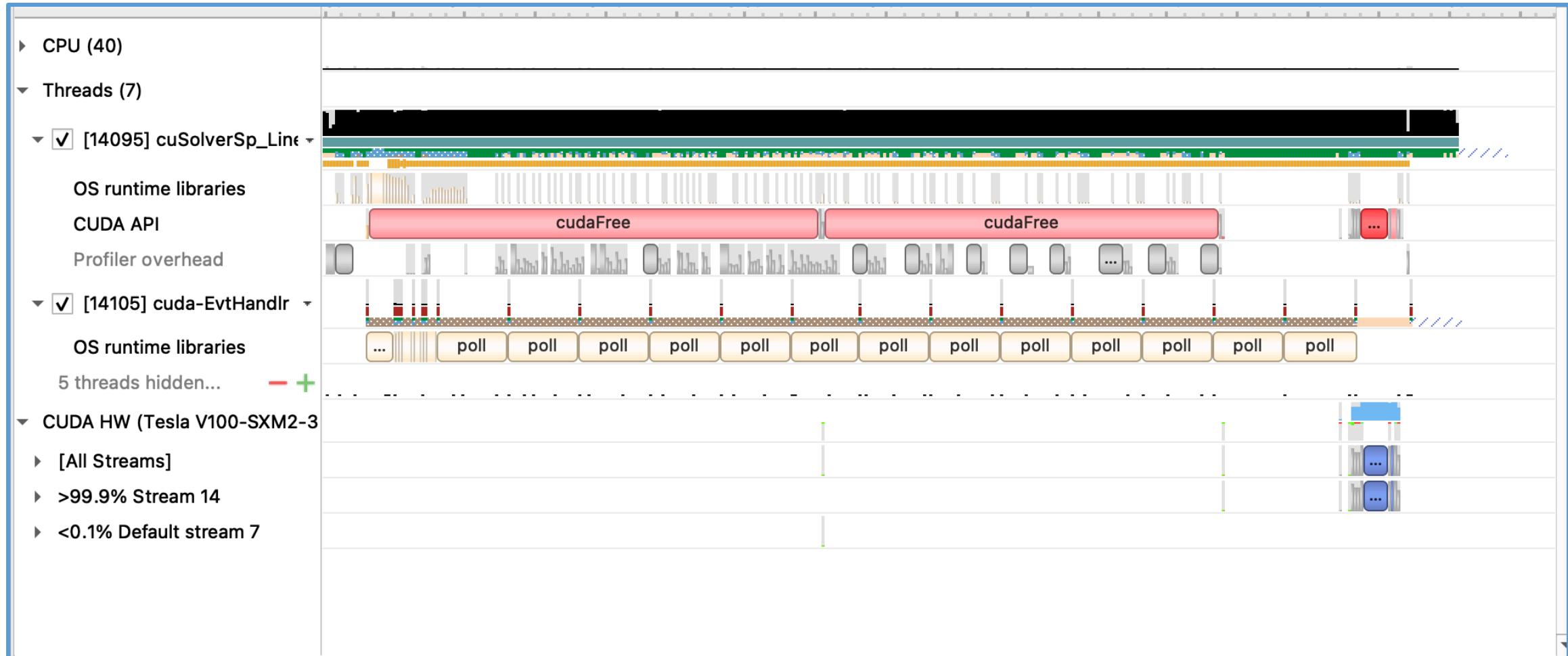
/home/mahidhar\_test/examples/7\_CUDALibraries/cuSolverSp\_LinearSolver/baseline.qdrep : File exists

Use `--force-overwrite true` to override existing files.

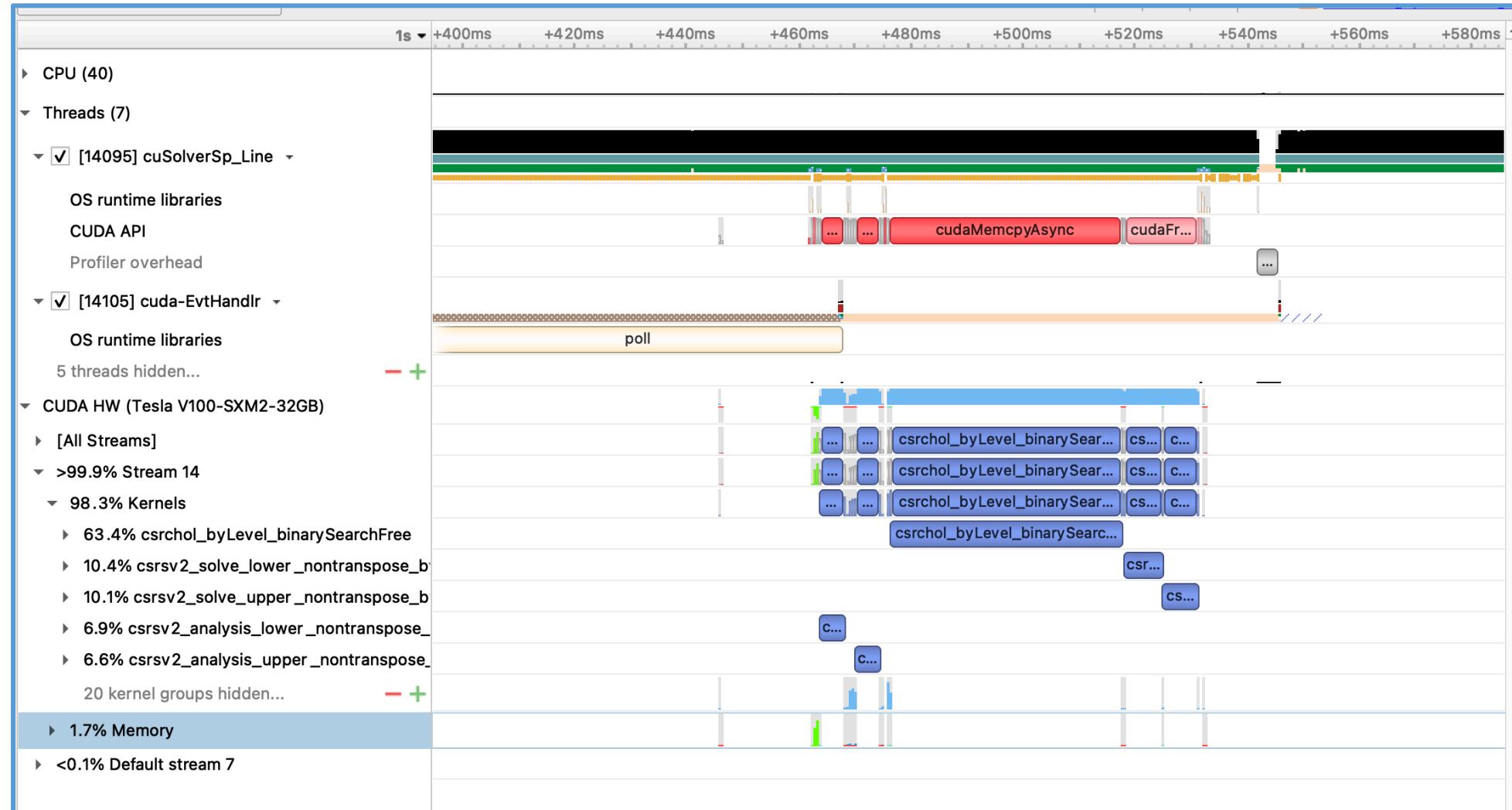
Please manually fetch report file(s) from:

\* /tmp/nsys-report-940c-3a9e-7049-6092.qdrep

# Nsight Systems: Simple Profiling Example



# Nsight Systems: Simple Profiling Example (zoomed)



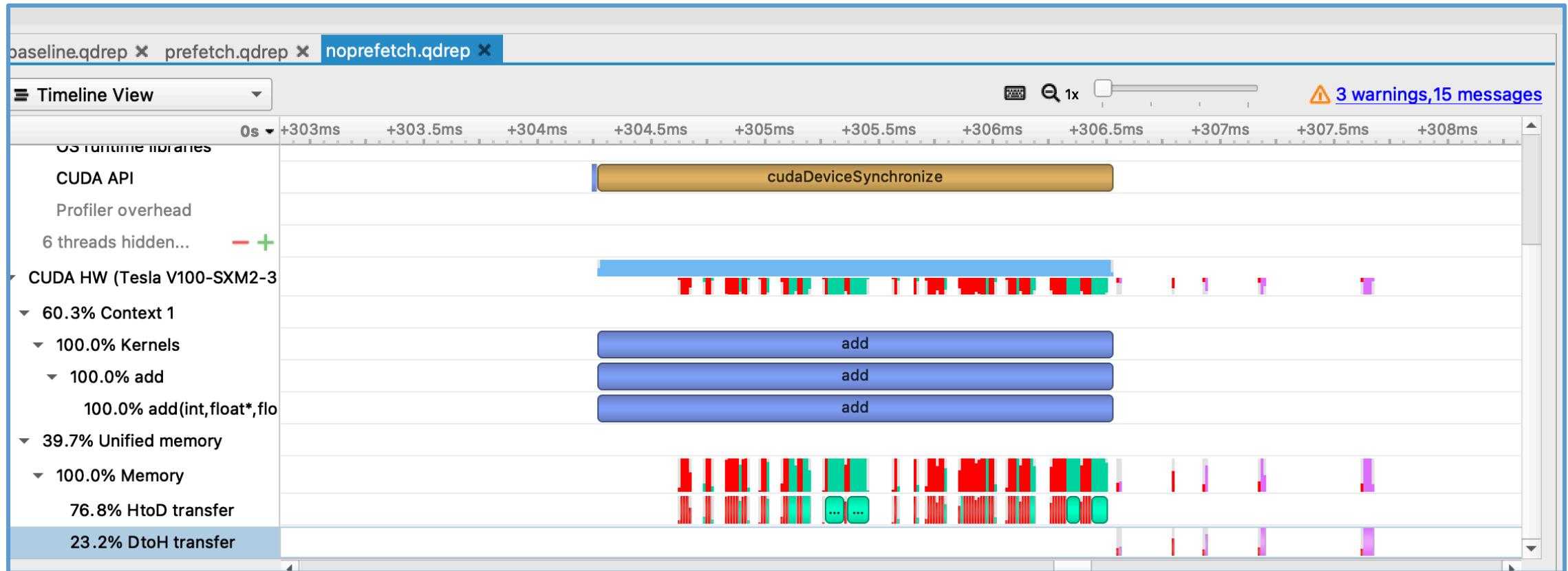
# Nsight Systems: Example illustrating process

- Reference:

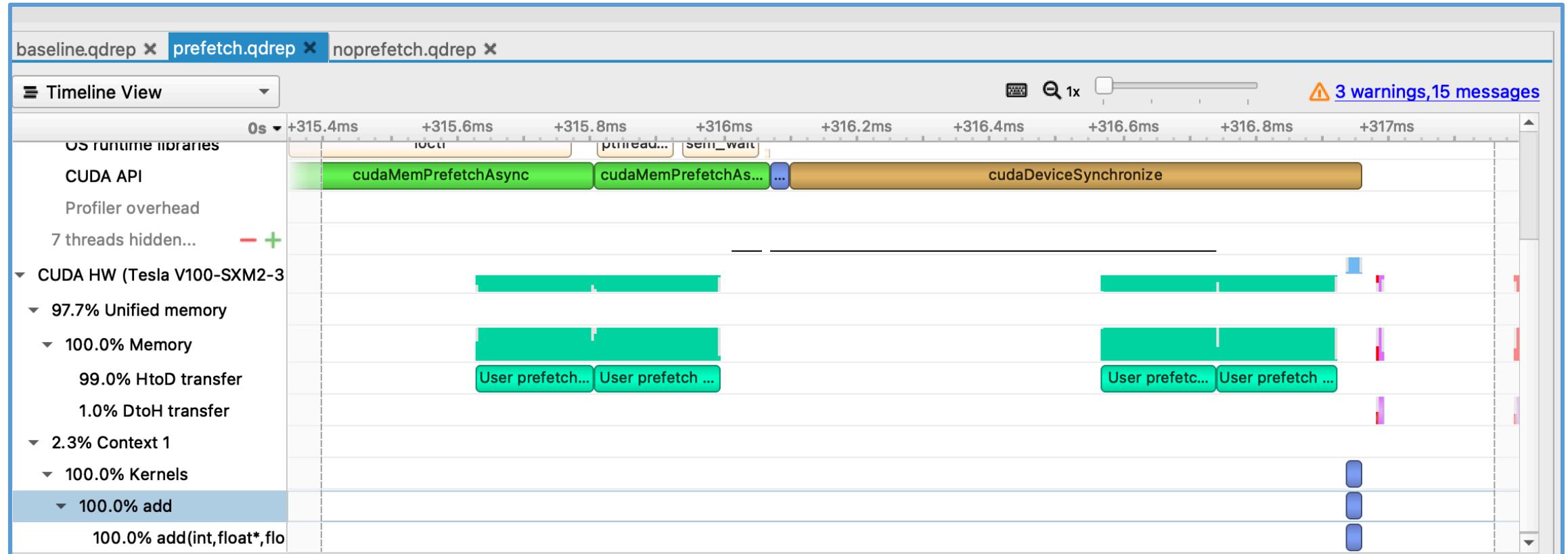
<https://developer.nvidia.com/blog/transitioning-nsight-systems-nvidia-visual-profiler-nvprof/>

- Sample code (in link above) uses unified memory data movement and does vector addition.
- Two approaches for the data movement are considered – with and without prefetch. We can run these on Expanse and look at the profiles.

# Nsight Systems: Example w/ no prefetch



# Nsight Systems: Example w/ prefetch



# Nsight Compute

- Interactive kernel profiler for CUDA applications. Part of CUDA toolkit
- Both graphical user interface and command line options.
- Features include:
  - Compare performance metrics between baselines and the current run,
  - CUDA Task Graph Profiling
  - Source code correlation
  - Roofline Analysis to visualize performance headroom
- Currently disabled on Expanse and Comet (restricted permissions due to security reasons)

Good example of optimization workflow here:

<https://developer.nvidia.com/blog/analysis-driven-optimization-preparing-for-analysis-with-nvidia-nsight-compute-part-1/>

# Summary

- Several command line and GUI based options available for profiling
- Some features limited by permissions restrictions (security considerations). Will be enabled in the future if nodes are exclusive access and with a feature request in SLURM.
- **nvprof** is the simplest profiling tool – command line approach with many different events, metrics options. CUDA/OpenACC codes can be easily profiled.
- **Nsight Systems** offers both CLI and GUI options to do system level profiling (both CPU and GPU components covered).
- **Nsight Compute** is a powerful Interactive kernel profiler for CUDA applications. *Currently disabled on Expanse, will be opened in the future.*