Profiling Celeritas

Peter Heywood, Research Software Engineer

The University of Sheffield

2024-03-27

Context

Increase Science Throughput

- Ever-increasing demand for increased simulation throughput

1. Buy more / “better” hardware
2. **Improve Software**
    - Improve implementations
    - Improve algorithms (i.e. work efficiency)

- Must **understand software performance** to improve performance

Profile

Profiling Tools

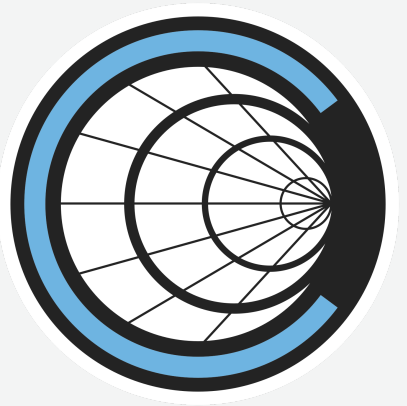
- CPU-only profilers
  - `gprof`, `perf`, Kcachegrind, VTune, …

- **NVIDIA Profiling tools**
  - **Nsight Systems**
  - **NVIDIA Nsight Compute**
  - `nvprof`

- AMD Profiling tools
  - `roctracer`
  - `rocsys`
  - `rocprofv2`

Celeritas

> The Celeritas project implements HEP detector physics on GPU accelerator hardware with the ultimate goal of supporting the massive computational requirements of the HL-LHC upgrade.



- github.com/celeritas-project/celeritas
- NVIDIA GPUs via CUDA
- AMD GPUs via HIP

- **Ben Morgan - “GPU workflows: Celeritas and AdePT simulation”**
- Ben Morgan - “Detector Simulations in Particle Physics”

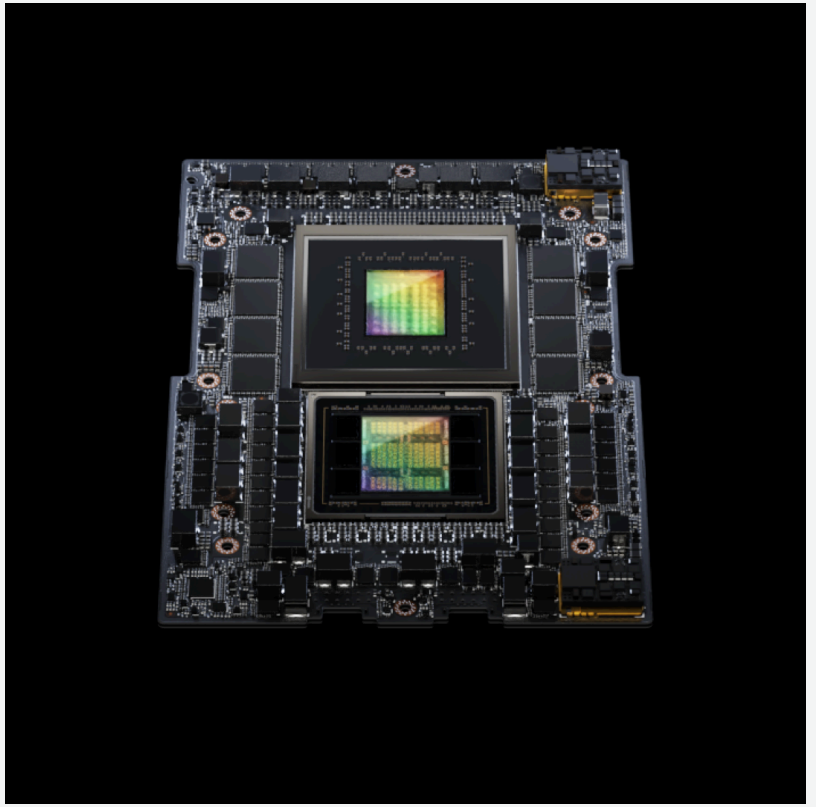
Graphics Processing Unit(s)

- Highly-parallel many-core co-processors
- Optimised for throughput
- (Relatively) Low volume of High-bandwidth memory
- Power efficient (for suitable workloads)
- Often connected via low-bandwidth PCIe

Titan Xp & Titan V GPUs

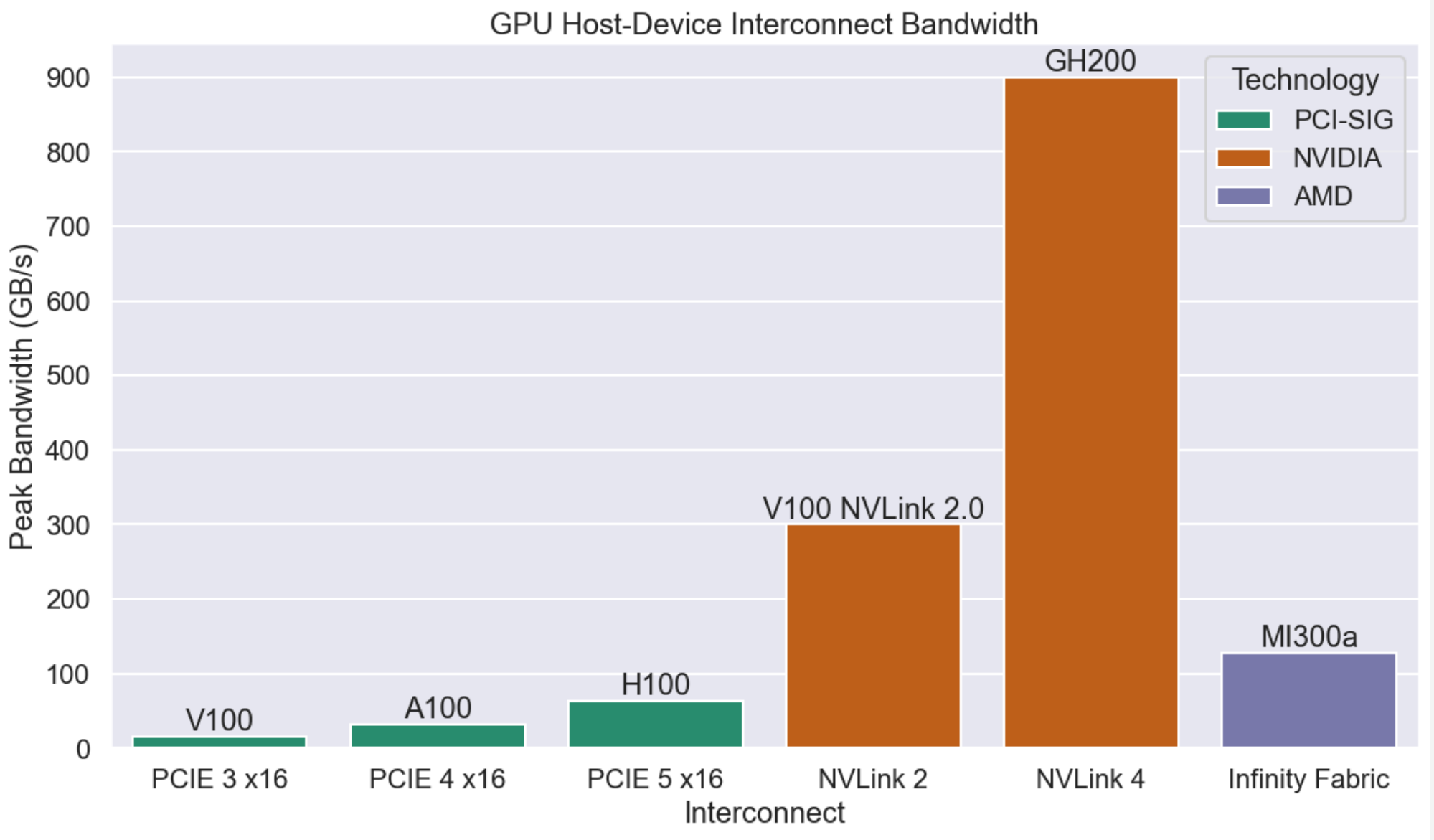
NVIDIA Grace Hopper Superchip

- GH200 480GB
  - **72-core ARM CPU**
  - 480GB LPDDR5X
  - H100 GPU (132 SMs)
  - 96GB HBM3e (4TB/s)
  - **NVLink-C2C 900 GB/s bidirectional bandwidth**
  - 450-1000W
- 3 now included in the Bede Tier 2 HPC facility



NVIDIA Grace Hopper Superchip

Host-Device Bandwidth



Celeritas test suite on GH200

```
$ ctest
# ...
99% tests passed, 2 tests failed out of 203
```

```
$ ctest --rerun-failed --output-on-failure
# ...
1/2 Test #158: celeritas/mat/Material ..........***Failed
    Error regular expression found in output. Regex=[tests FAILED]  0.
# ...
2/2 Test #160: celeritas/phys/Particle .........   Passed    0.61 sec

50% tests passed, 1 tests failed out of 2
```

JSON Comparison	`mass_radiation_coeff`
Expected	0.0360539283945309
Actual	0.0360539283945531

Profiling Celeritas

Inputs / Configuration

- Inputs should ideally be:
  - Representative of real-world use
  - Large enough to fully utilise hardware
  - Small enough to generate usable profile data
- Optimised build
  - `-DCMAKE_BUILD_TYPE=Release`, `-O3 -lineinfo`
- Celeritas `v0.4.2` with VecGeom `v1.2.4`

Profiling Scenario

- `cms2018+field+msc` from celeritas-project/regression
- `celer-sim`

- 16 Events
- 1300 primaries per event
- 1048576 track slots (max threads)

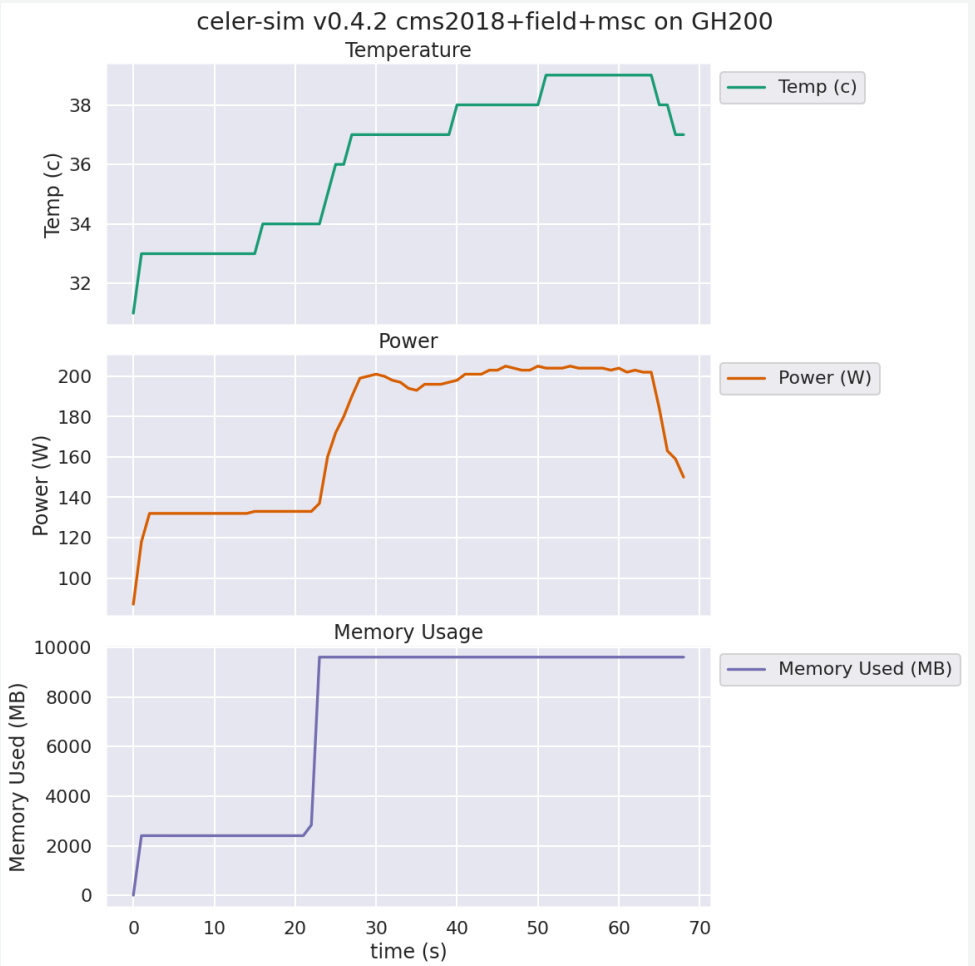
```
$ time ./bin/celer-sim cms2018+field+msc
# ...
real    1m13.997s
user    1m7.096s
sys     0m0.871s
```


```
{
    "_exe": "celer-sim",
    "_format": "celer-sim",
    "_geometry": "vecgeom",
    "_instance": 0,
    "_name": [
        "cms2018+field+msc",
        "vecgeom",
        "gpu"
    ],
    "_outdir": "cms2018+field+msc-vecgeom-gpu",
    "_timeout": 600.0,
    "_use_celeritas": true,
    "_version": "0.4.2",
    "action_diagnostic": false,
    "brem_combined": false,
    "cuda_heap_size": null,
    "cuda_stack_size": 8192,
    "default_stream": false,
    "environ": {},
    "event_file": null,
    "field": [
        0.0,
        0.0,
        1.0
    ],
    "field_options": {
        "delta_chord": 0.025,
        "delta_intersection": 1e-05,
        "epsilon_rel_max": 0.001,
        "epsilon_step": 1e-05,
        "errcon": 0.0001,
```

Power Consumption Monitoring

- Can use `nvml` to monitor GPU resource consumption
- Does not account for the rest of the system
- github.com/willfurnass/gpuutiliz
- Peak (per second) consumption only 200W / 900W

```
1 gpuutiliz -frequency 1 &
2 gupid=$!
3 ./bin/celer-sim input.json
4 kill ${gupid}
```



Nsight Systems

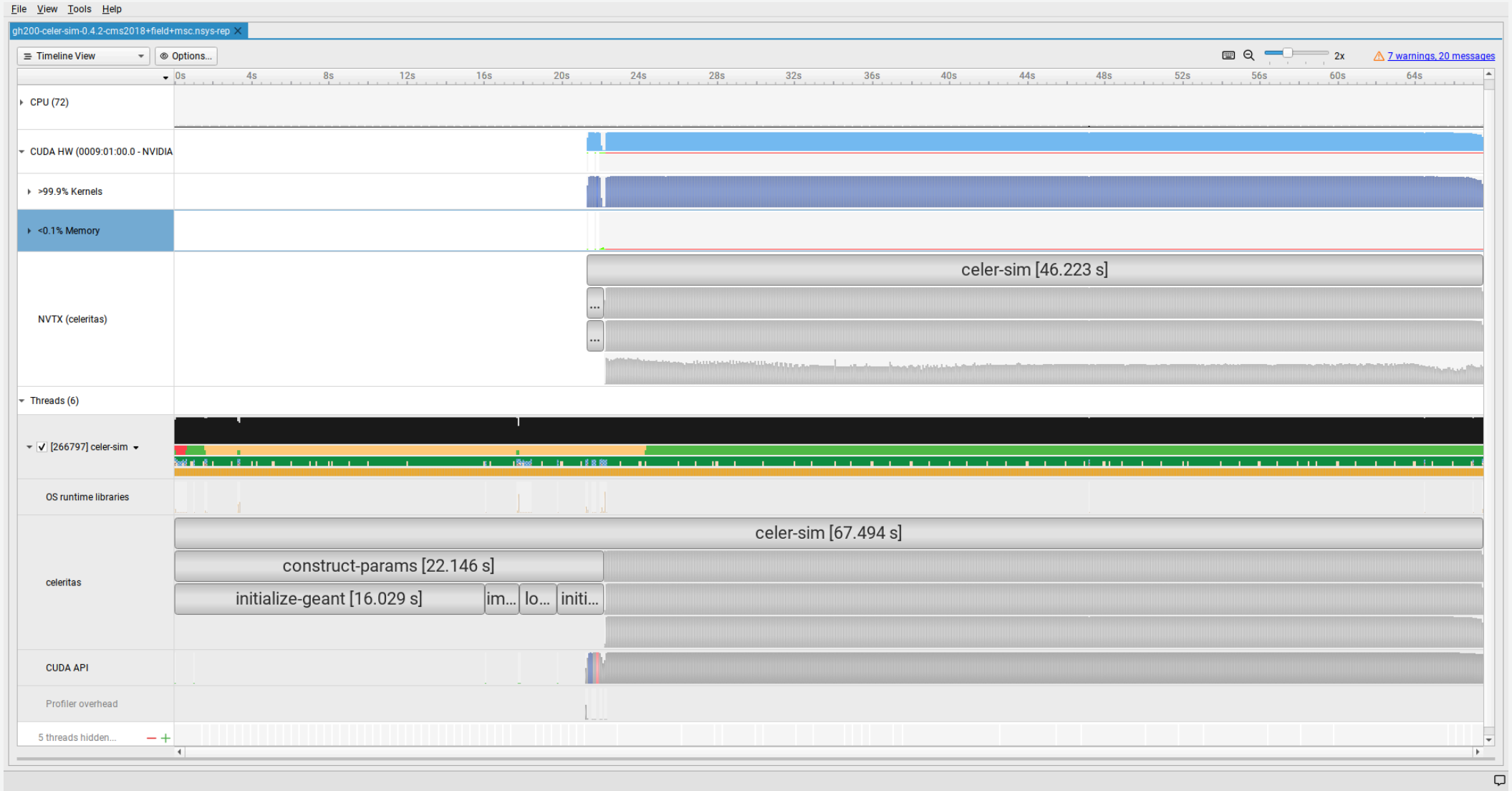
Nsight Systems

- System-wide performance analysis
- CPU + GPU
- Visualise a timeline of events
- CUDA API information, kernel block sizes, etc
- Pascal GPUs or newer (SM 60+)

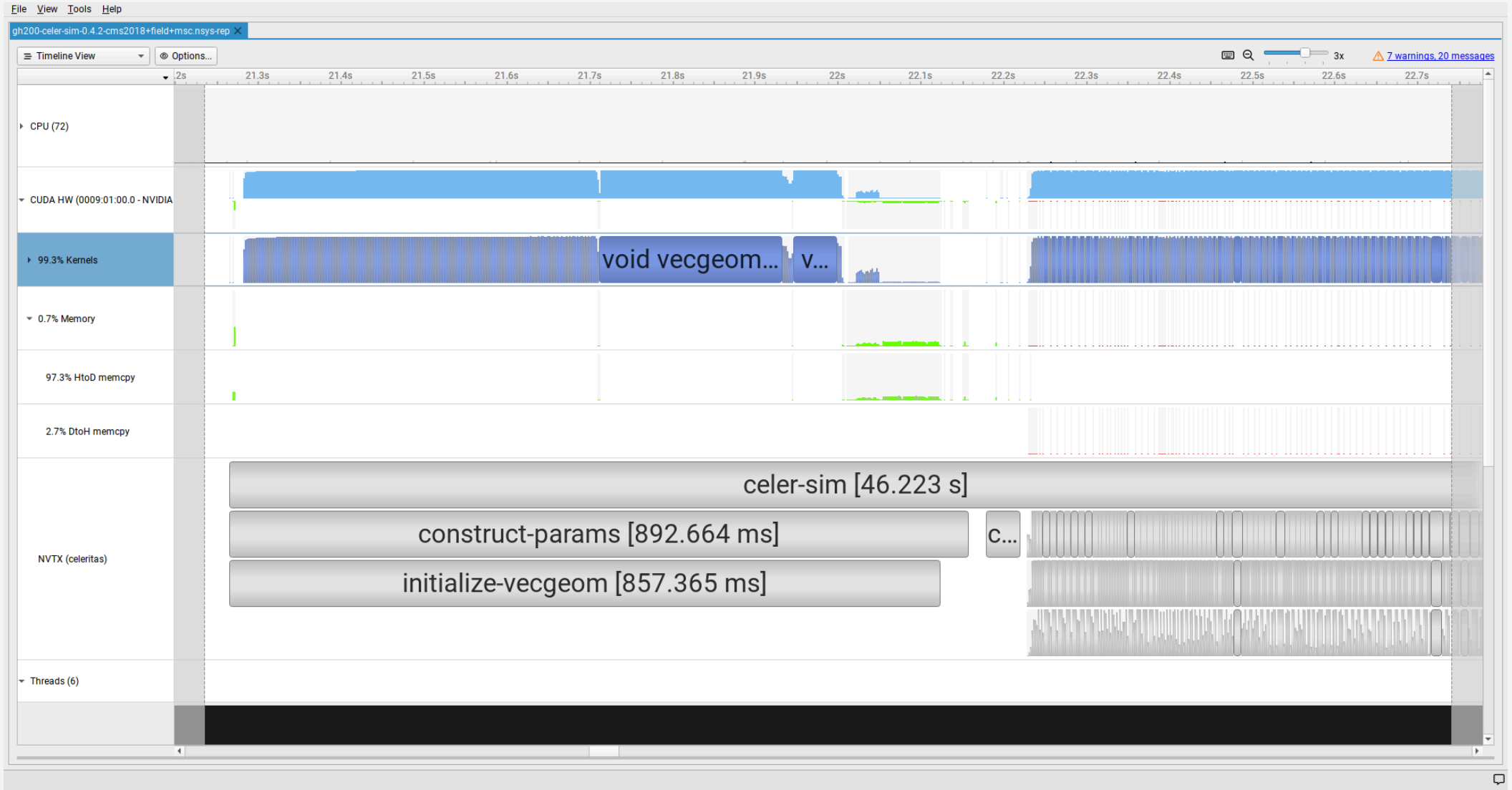
```
1 nsys profile -o timeline ./bin/celer-sim input.json
2 nsys-ui timeline.nsys-rep
```

- Enable NVTX in Celeritas via `CELER_ENABLE_PROFILING=1`

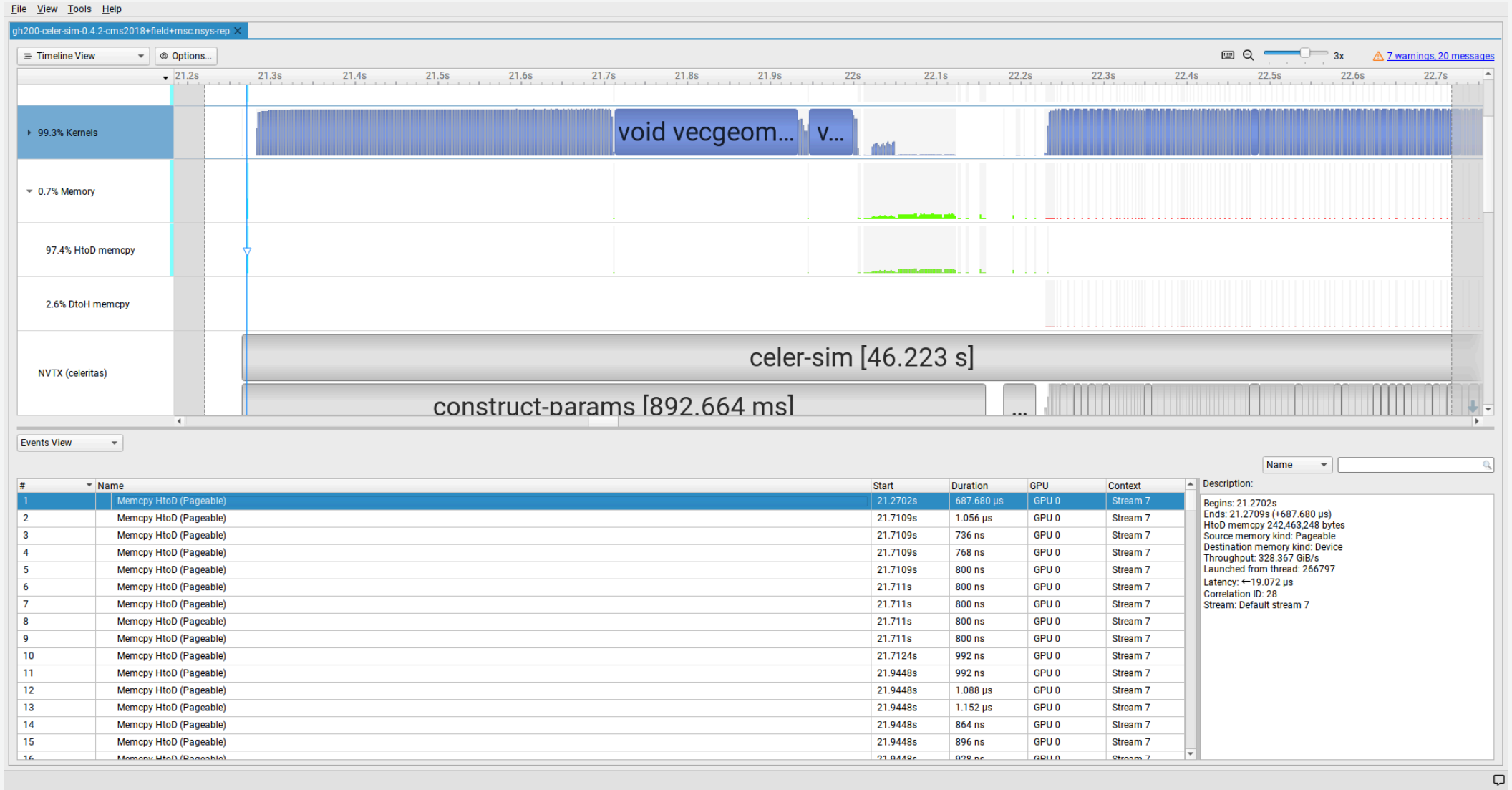
nsys: Timeline



nsys: Host-Device Communication

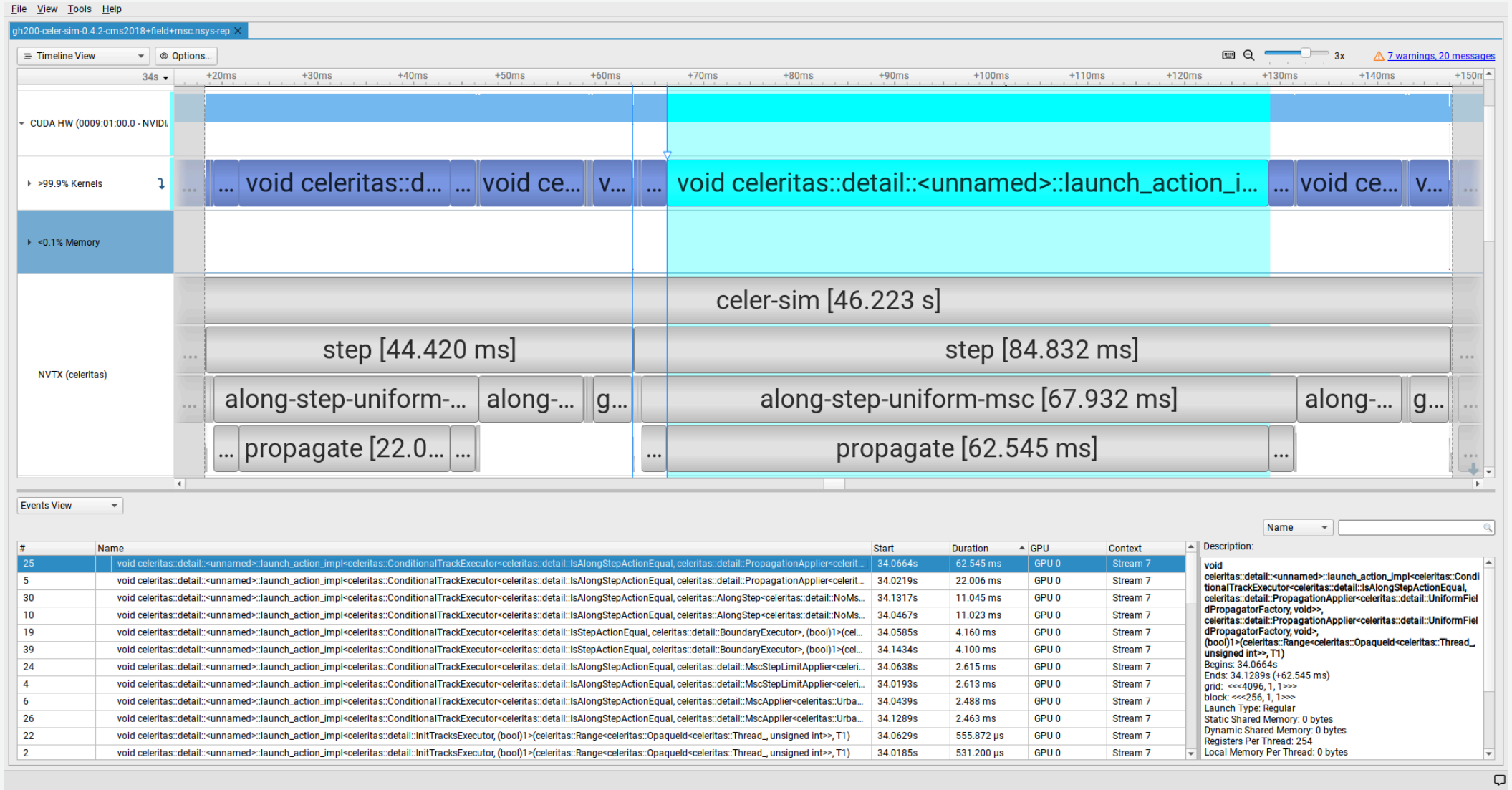


nsys: Host-Device Communication



242MB, 690µs @ 328GB/s

nsys: Longest Duration Kernel



#	Name	Start	Duration	GPU	Context
25	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepActionEqual, celeritas::detail::PropagationApplier<celerit...	34.0664s	62.545 ms	GPU 0	Stream 7
5	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepActionEqual, celeritas::detail::PropagationApplier<celerit...	34.0219s	22.006 ms	GPU 0	Stream 7
30	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepActionEqual, celeritas::AlongStep<celeritas::detail::NoMs...	34.1317s	11.045 ms	GPU 0	Stream 7
10	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepActionEqual, celeritas::AlongStep<celeritas::detail::NoMs...	34.0467s	11.023 ms	GPU 0	Stream 7
19	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsStepActionEqual, celeritas::detail::BoundaryExecutor>, (bool)1>(cel...	34.0585s	4.160 ms	GPU 0	Stream 7
39	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsStepActionEqual, celeritas::detail::BoundaryExecutor>, (bool)1>(cel...	34.1434s	4.100 ms	GPU 0	Stream 7
24	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepActionEqual, celeritas::detail::MscStepLimitApplier<celeri...	34.0638s	2.615 ms	GPU 0	Stream 7
4	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepActionEqual, celeritas::detail::MscStepLimitApplier<celeri...	34.0193s	2.613 ms	GPU 0	Stream 7
6	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepActionEqual, celeritas::detail::MscApplier<celeritas::Urba...	34.0439s	2.488 ms	GPU 0	Stream 7
26	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepActionEqual, celeritas::detail::MscApplier<celeritas::Urba...	34.1289s	2.463 ms	GPU 0	Stream 7
22	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::detail::InitTracksExecutor, (bool)1>(celeritas::Range<celeritas::OpaqueId<celeritas::Thread_, unsigned int>>, T1)	34.0629s	555.872 µs	GPU 0	Stream 7
2	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::detail::InitTracksExecutor, (bool)1>(celeritas::Range<celeritas::OpaqueId<celeritas::Thread_, unsigned int>>, T1)	34.0185s	531.200 µs	GPU 0	Stream 7

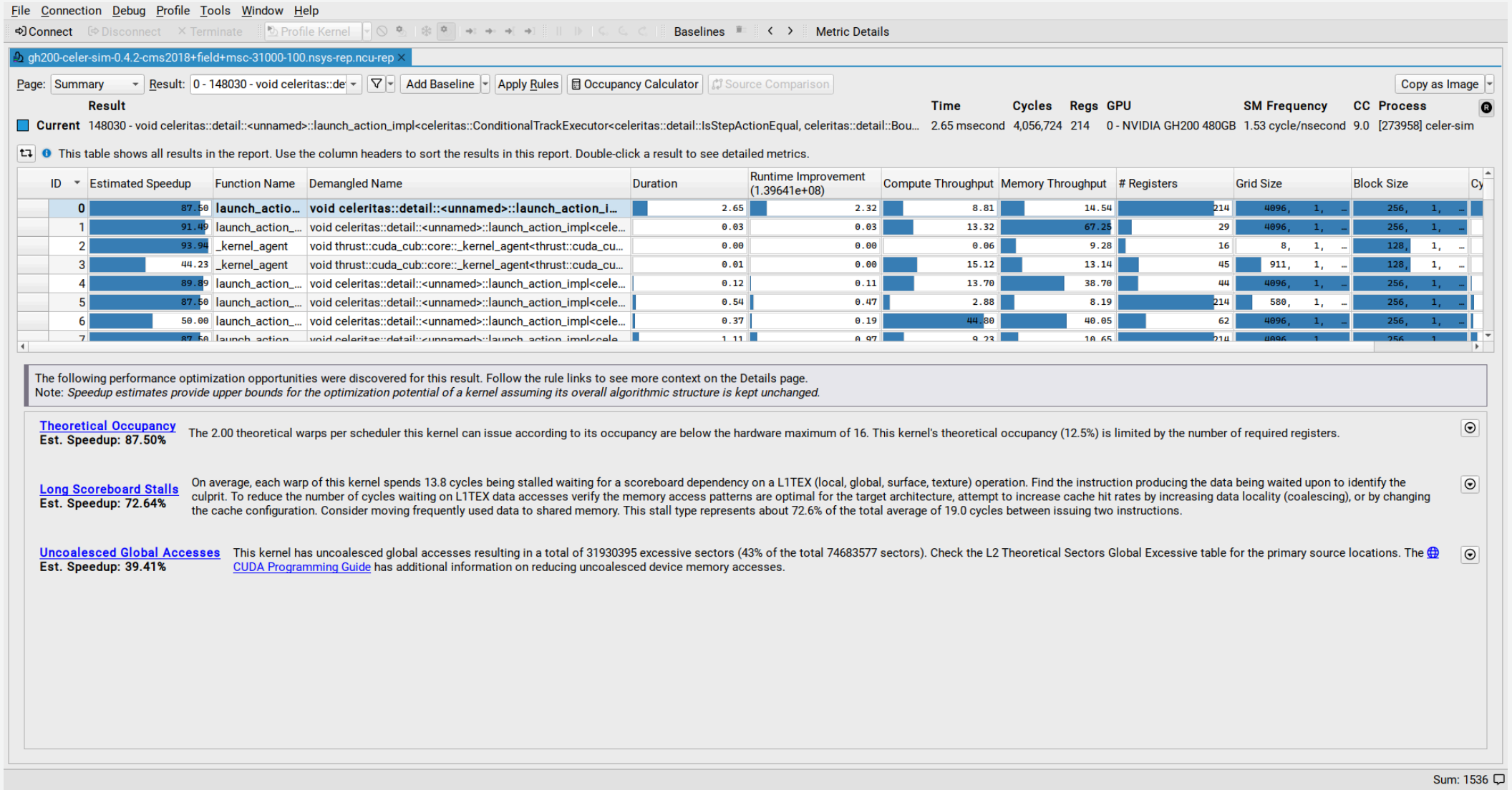
Nsight Compute

Nsight Compute

- Detailed GPU performance metrics
- Compile with `-lineinfo` for line-level profiling
- Use `--set=full` for non-interactive profiling
- Replays GPU kernels many times - **significant runtime increase**
- Reduce captured kernels via filtering, `-s`, `-c` etc.
- Volta+ (SM >= 70)

```
1 # All metrics, skip 64 kernels, capture 128.
2 ncu --set=full -s 64 -c 128 -o metrics.ncu-rep \
3     ./bin/celer-sim input.json
4 ncu-ui metrics.ncu-rep
```

ncu: Summary


File Connection Debug Profile Tools Window Help
Connect Disconnect Terminate Profile Kernel Baselines Metric Details
gh200-celer-sim-0.4.2-cms2018+field+msc-31000-100.nsys-rep.ncu-rep
Page: Summary Result: 0 - 148030 - void celeritas::de Add Baseline Apply Rules Occupancy Calculator Source Comparison Copy as Image
Result Time Cycles Regs GPU SM Frequency CC Process
Current 148030 - void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsStepActionEqual, celeritas::detail::Bou... 2.65 msecond 4,056,724 214 0 - NVIDIA GH200 480GB 1.53 cycle/nsecond 9.0 [273958] celer-sim
This table shows all results in the report. Use the column headers to sort the results in this report. Double-click a result to see detailed metrics.
ID Estimated Speedup Function Name Demangled Name Duration Runtime Improvement (1.39641e+08) Compute Throughput Memory Throughput # Registers Grid Size Block Size
0 87.50 launch_actio... void celeritas::detail::<unnamed>::launch_action_i... 2.65 2.32 8.81 14.54 214 4096, 1, ... 256, 1, ...
1 91.49 launch_action_... void celeritas::detail::<unnamed>::launch_action_impl<cele... 0.03 0.03 13.32 67.25 29 4096, 1, ... 256, 1, ...
2 93.94 _kernel_agent void thrust::cuda_cub::core::_kernel_agent<thrust::cuda_cu... 0.00 0.00 0.06 9.28 16 8, 1, ... 128, 1, ...
3 44.23 _kernel_agent void thrust::cuda_cub::core::_kernel_agent<thrust::cuda_cu... 0.01 0.00 15.12 13.14 45 911, 1, ... 128, 1, ...
4 89.89 launch_action_... void celeritas::detail::<unnamed>::launch_action_impl<cele... 0.12 0.11 13.70 38.70 44 4096, 1, ... 256, 1, ...
5 87.50 launch_action_... void celeritas::detail::<unnamed>::launch_action_impl<cele... 0.54 0.47 2.88 8.19 214 580, 1, ... 256, 1, ...
6 50.00 launch_action_... void celeritas::detail::<unnamed>::launch_action_impl<cele... 0.37 0.19 44.80 40.05 62 4096, 1, ... 256, 1, ...
The following performance optimization opportunities were discovered for this result. Follow the rule links to see more context on the Details page.
Note: Speedup estimates provide upper bounds for the optimization potential of a kernel assuming its overall algorithmic structure is kept unchanged.
Theoretical Occupancy
Est. Speedup: 87.50%
The 2.00 theoretical warps per scheduler this kernel can issue according to its occupancy are below the hardware maximum of 16. This kernel's theoretical occupancy (12.5%) is limited by the number of required registers.
Long Scoreboard Stalls
Est. Speedup: 72.64%
On average, each warp of this kernel spends 13.8 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 72.6% of the total average of 19.0 cycles between issuing two instructions.
Uncoalesced Global Accesses
Est. Speedup: 39.41%
This kernel has uncoalesced global accesses resulting in a total of 31930395 excessive sectors (43% of the total 74683577 sectors). Check the L2 Theoretical Sectors Global Excessive table for the primary source locations. The CUDA Programming Guide has additional information on reducing uncoalesced device memory accesses.
Sum: 1536

ncu: “Speed of Light”

File Connection Debug Profile Tools Window Help

Connect | Disconnect | Terminate | Profile Kernel | Baselines | Metric Details

gh200-celer-sim-0.4.2-cms2018+field+msc-slowest.ncu-rep

Page: Details | Result: 0 - 94274 - void celeritas::de | Add Baseline | Apply Rules | Occupancy Calculator | Source Comparison | Copy as Image

Result		Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	94274 - void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepA...	86.57 msecond	125,999,443	254	0 - NVIDIA GH200 480GB	1.46 cycle/nsecond	9.0	[290400] celer-sim

GPU Speed Of Light Throughput

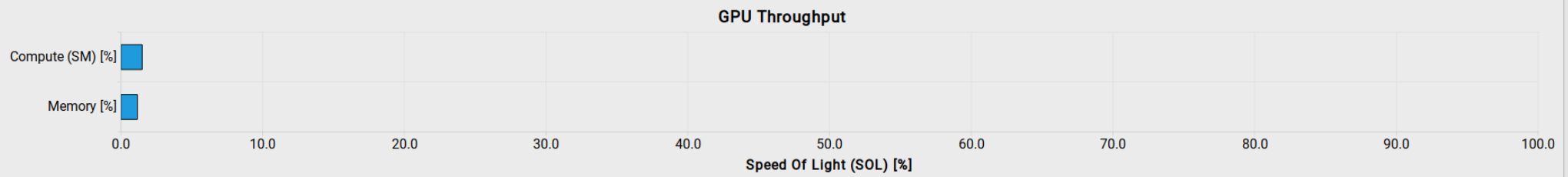
GPU Throughput Chart

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

Compute (SM) Throughput [%]	1.50	Duration [msecond]	86.57
Memory Throughput [%]	1.16	Elapsed Cycles [cycle]	125,999,443
L1/TEX Cache Throughput [%]	4.93	SM Active Cycles [cycle]	29,675,643.55
L2 Cache Throughput [%]	0.92	SM Frequency [cycle/nsecond]	1.46
DRAM Throughput [%]	0.09	DRAM Frequency [cycle/nsecond]	2.49

Latency Issue This kernel exhibits low compute throughput and memory bandwidth utilization relative to the peak performance of this device. Achieved compute throughput and/or memory bandwidth below 60.0% of peak typically indicate latency issues. Look at Scheduler Statistics and Warp State Statistics for potential reasons.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 2:1. The kernel achieved close to 0% of this device's fp32 peak performance and close to 0% of its fp64 peak performance. See the Kernel Profiling Guide for more details on roofline analysis.



PM Sampling

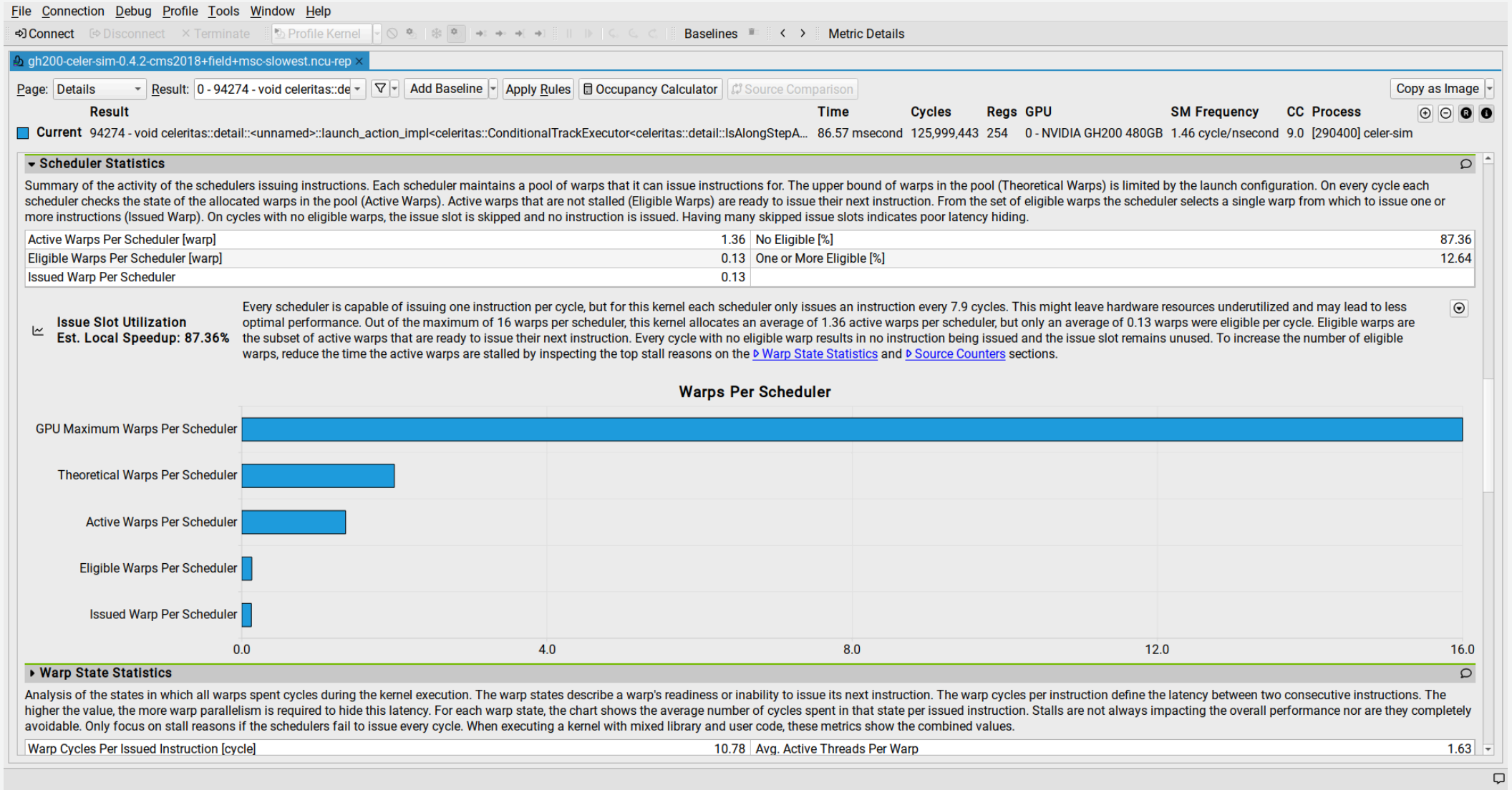
Timeline view of PM metrics sampled periodically over the workload duration. Data is collected across multiple passes. Use this section to understand workload behavior changes over its runtime.

Compute Workload Analysis

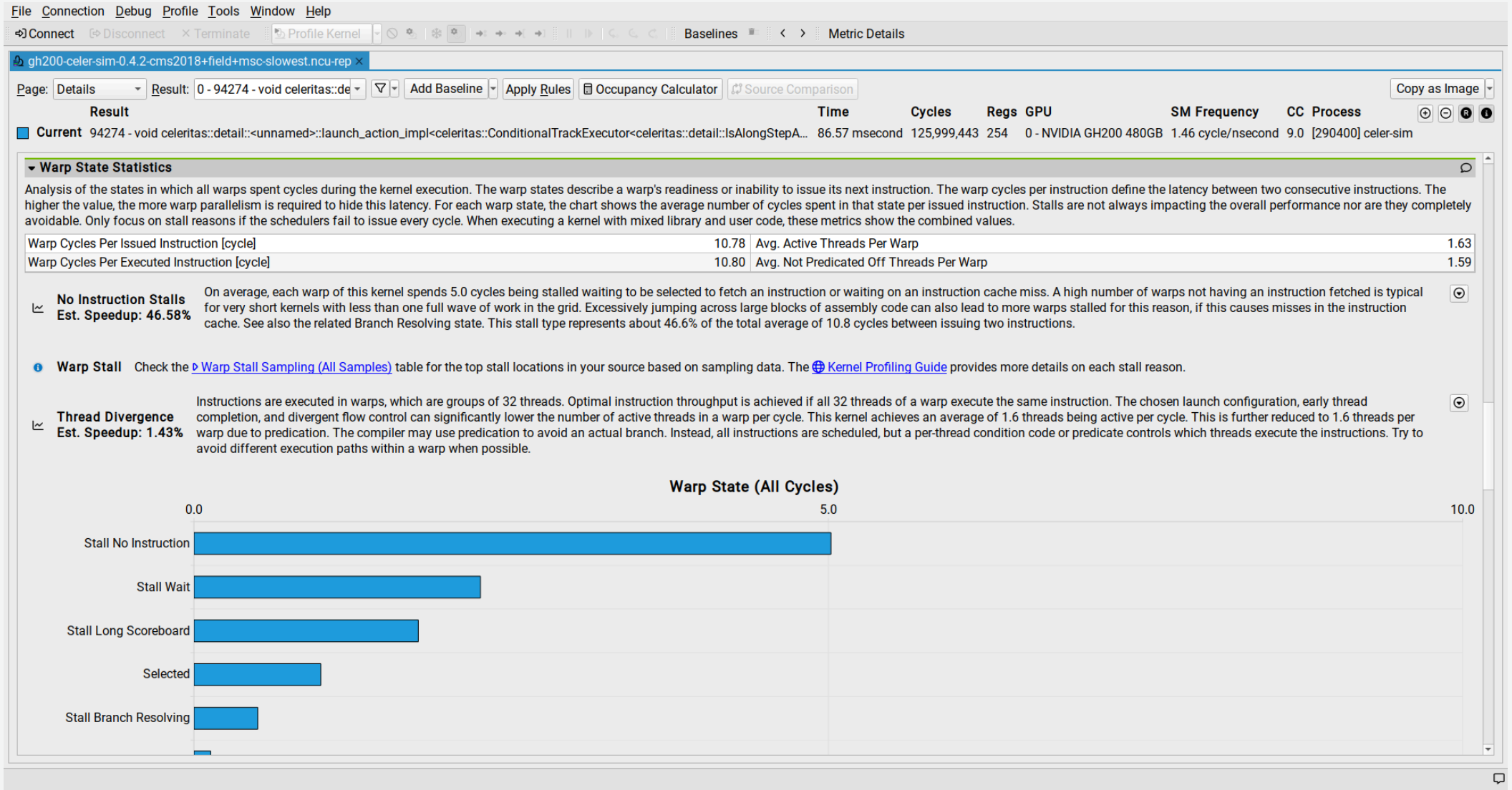
Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.

Executed Ipc Elapsed [inst/cycle]	0.06	SM Busy [%]	6.38
Executed Ipc Active [inst/cycle]	0.25	Issue Slots Busy [%]	6.38
Issued Ipc Active [inst/cycle]	0.26		

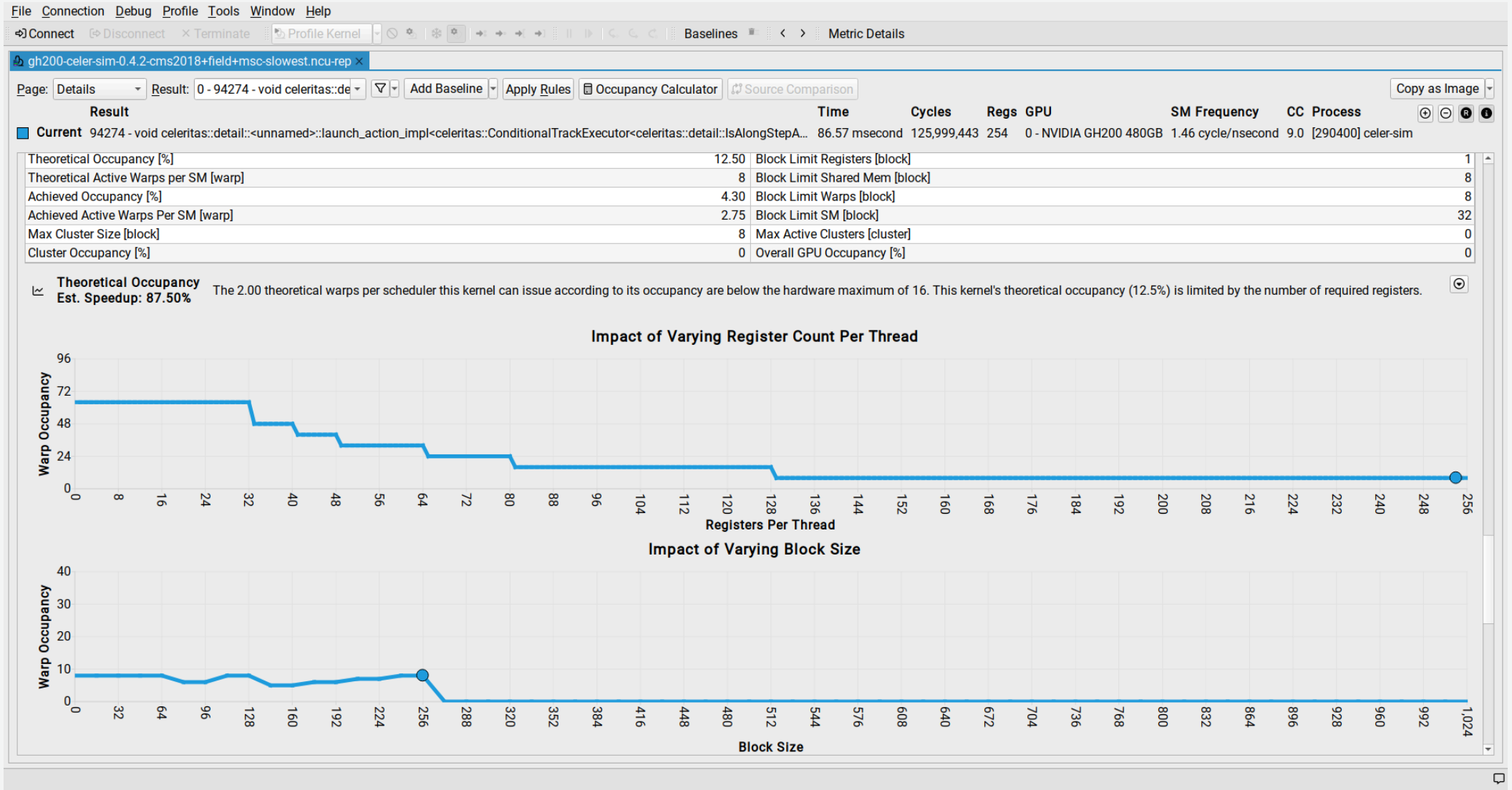
ncu: Scheduler



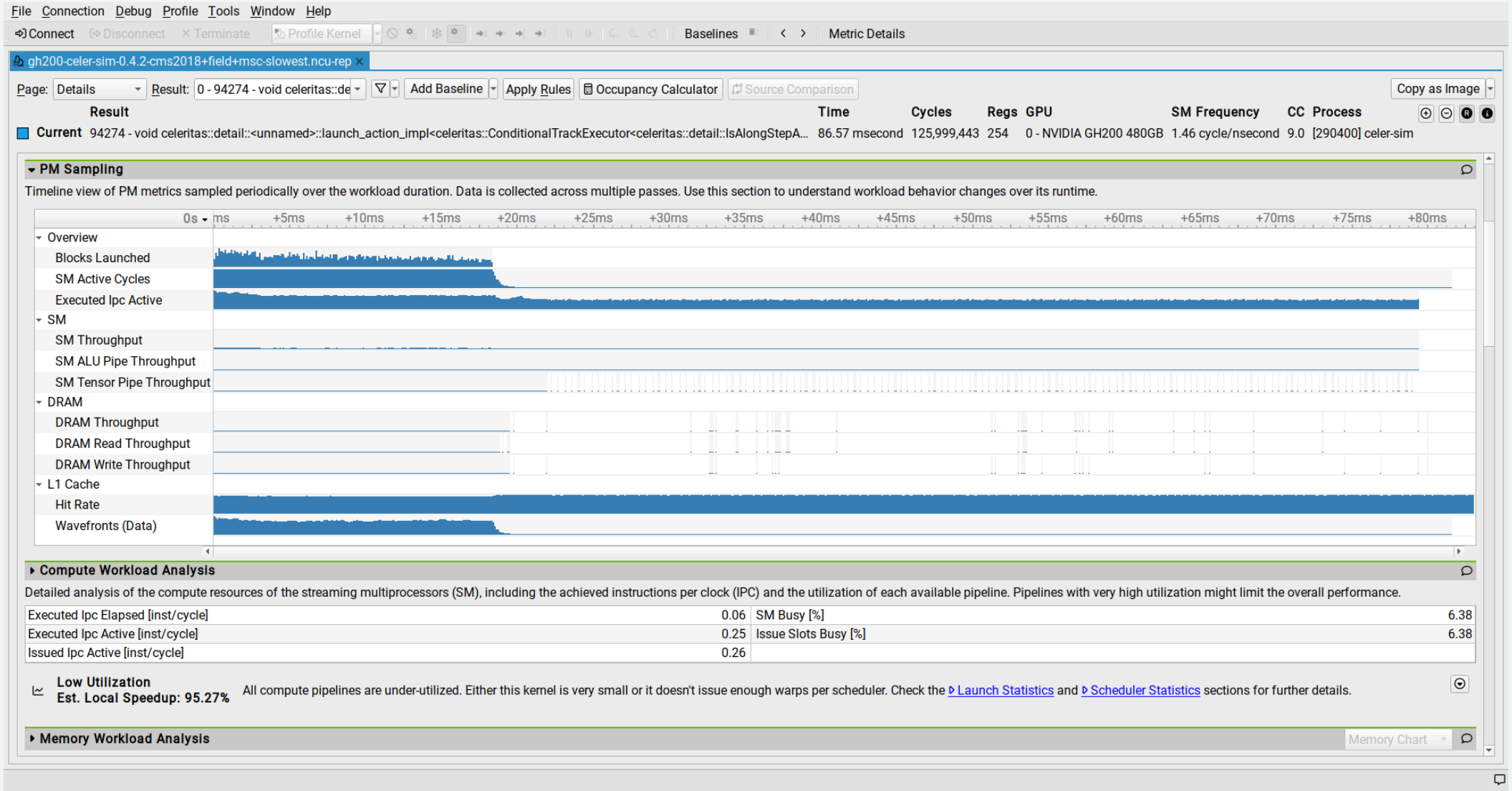
ncu: Warp state



ncu: Occupancy

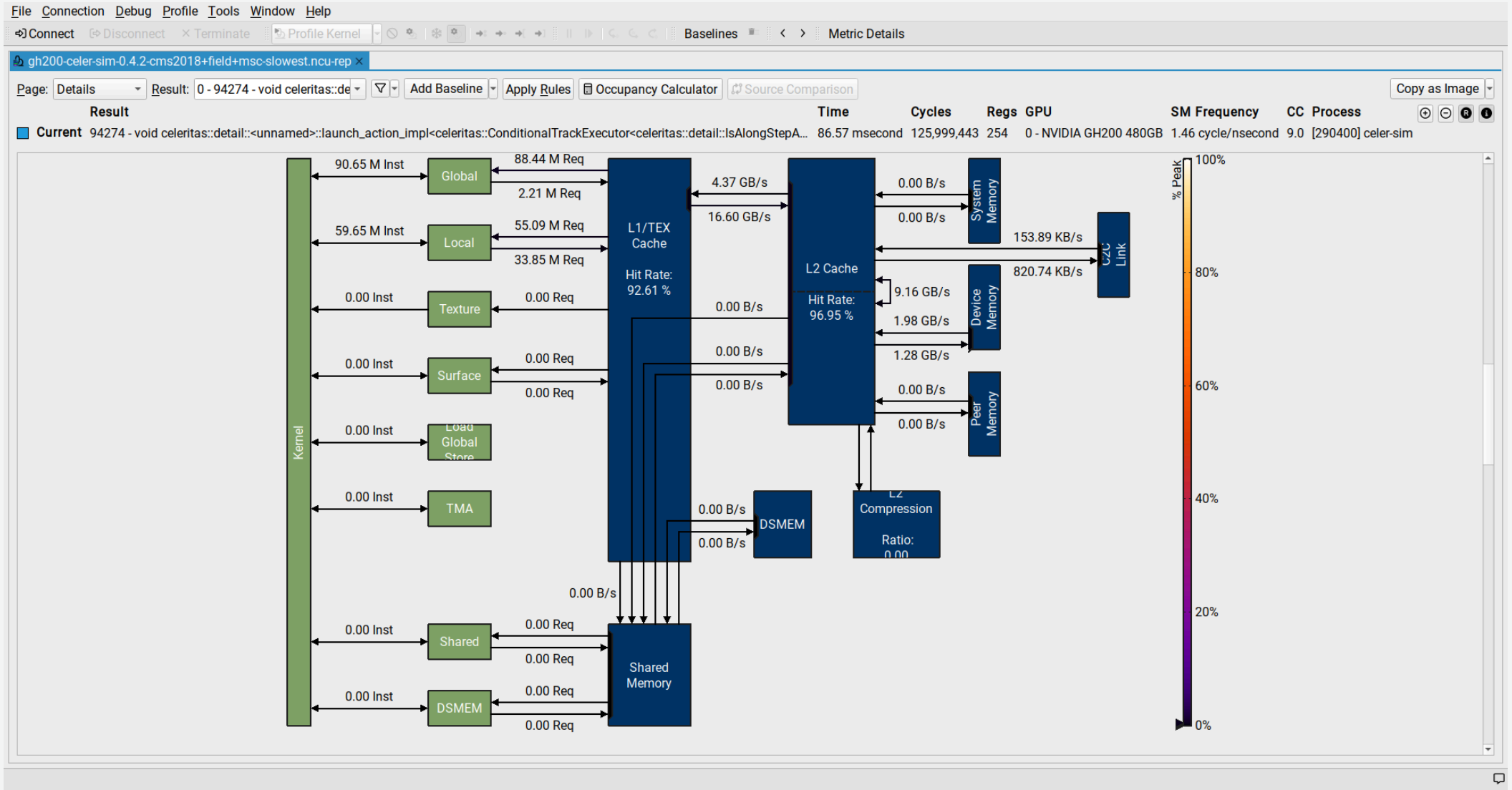


ncu: Performance Monitor Sampling



- Nsight Compute >= `2023.3` (distributed with CUDA 12.3)

ncu: Memory Access Pattern



Thank you

UKRI “Shaping the Future of UK large-scale compute” survey closes **29 March 2024** (this Friday!)

https://engagementhub.ukri.org/ukri-infrastructure/shaping-the-future-of-uk-large-scale-compute/

Additional Slides

Building Celeritas on GH200

- Some warnings which can be suppressed via `-Wno-psabi`
- GCC >= 10.1 on `aarch64`

```
include/VecGeom/base/Transformation3D.h: In member function
  ‘vecgeom::cxx::Vector3D<double>
  vecgeom::cxx::Transformation3D::Translation() const’:
include/VecGeom/base/Transformation3D.h:213:3: note: parameter
  passing for argument of type ‘vecgeom::cxx::Vector3D<double>’
  when C++17 is enabled changed to match C++14 in GCC 10.1
  213 |   {
      |   ^
```

Celeritas test suite on GH200

```
$ ctest
# ...
99% tests passed, 2 tests failed out of 203

Label Time Summary:
app            = 108.80 sec*proc (11 tests)
gpu            = 101.33 sec*proc (43 tests)
nomemcheck     = 107.88 sec*proc (9 tests)
unit           =  33.99 sec*proc (191 tests)

Total Test time (real) = 140.78 sec

The following tests FAILED:
        158 - celeritas/mat/Material (Failed)
        160 - celeritas/phys/Particle (SEGFAULT)
```

Celeritas test suite on GH200

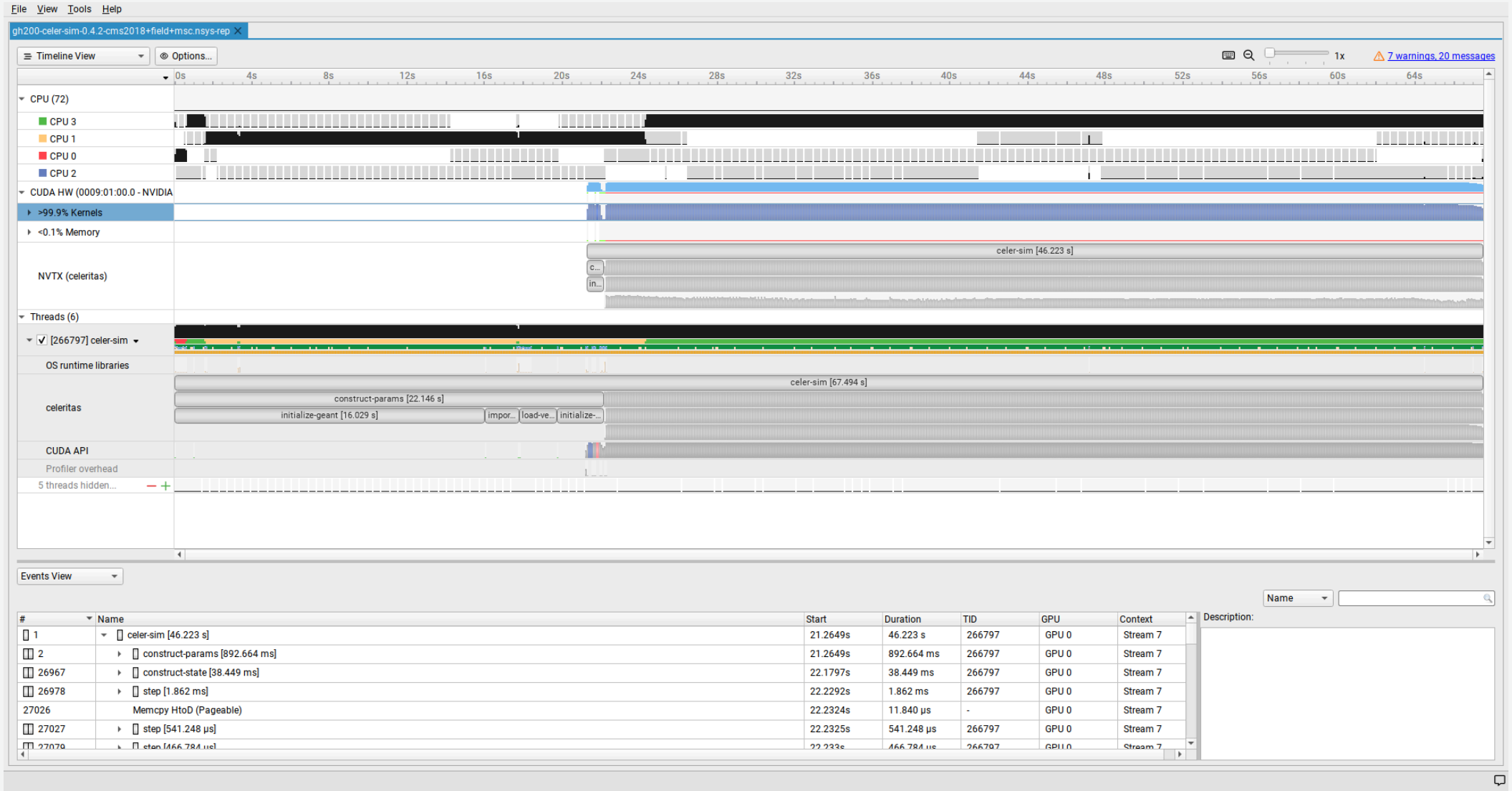
```
$ ctest --rerun-failed --output-on-failure
# ...
1/2 Test #158: celeritas/mat/Material ..........***Failed
    Error regular expression found in output. Regex=[tests FAILED]  0.
# ...
2/2 Test #160: celeritas/phys/Particle .........   Passed    0.61 sec

50% tests passed, 1 tests failed out of 2

The following tests FAILED:
        158 - celeritas/mat/Material (Failed)
```

JSON Comparison	`mass_radiation_coeff`
Expected	0.0360539283945309
Actual	0.0360539283945531

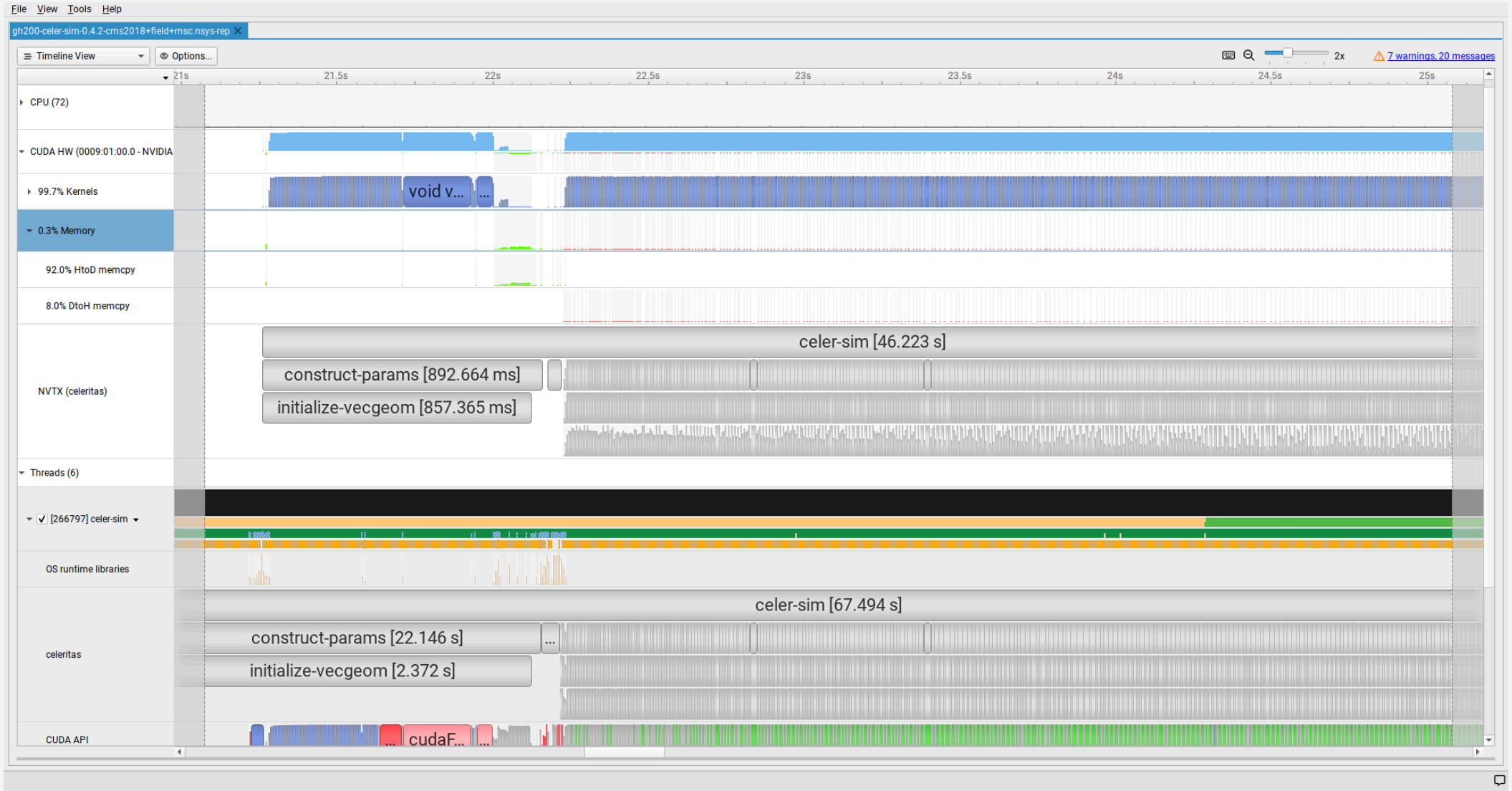
Running the scenario

```
$ time ./bin/celer-sim cms2018+field+msc.json
status: Loading input and initializing problem data
status: Initializing Geant4 run manager
status: Initializing Geant4 geometry
info: Loading Geant4 geometry from GDML at /path/to/cms2018.gdml
status: Building Geant4 physics tables
status: Transferring data from Geant4
status: Loading external elemental data
status: Loading VecGeom geometry from GDML at /path/to/cms2018.gdml
status: Initializing tracking information
celeritas/src/celeritas/geo/GeoMaterialParams.cc:205: warning: Some ge

# ...

real    1m13.997s
user    1m7.096s
sys     0m0.871s
```

nsys: Timeline



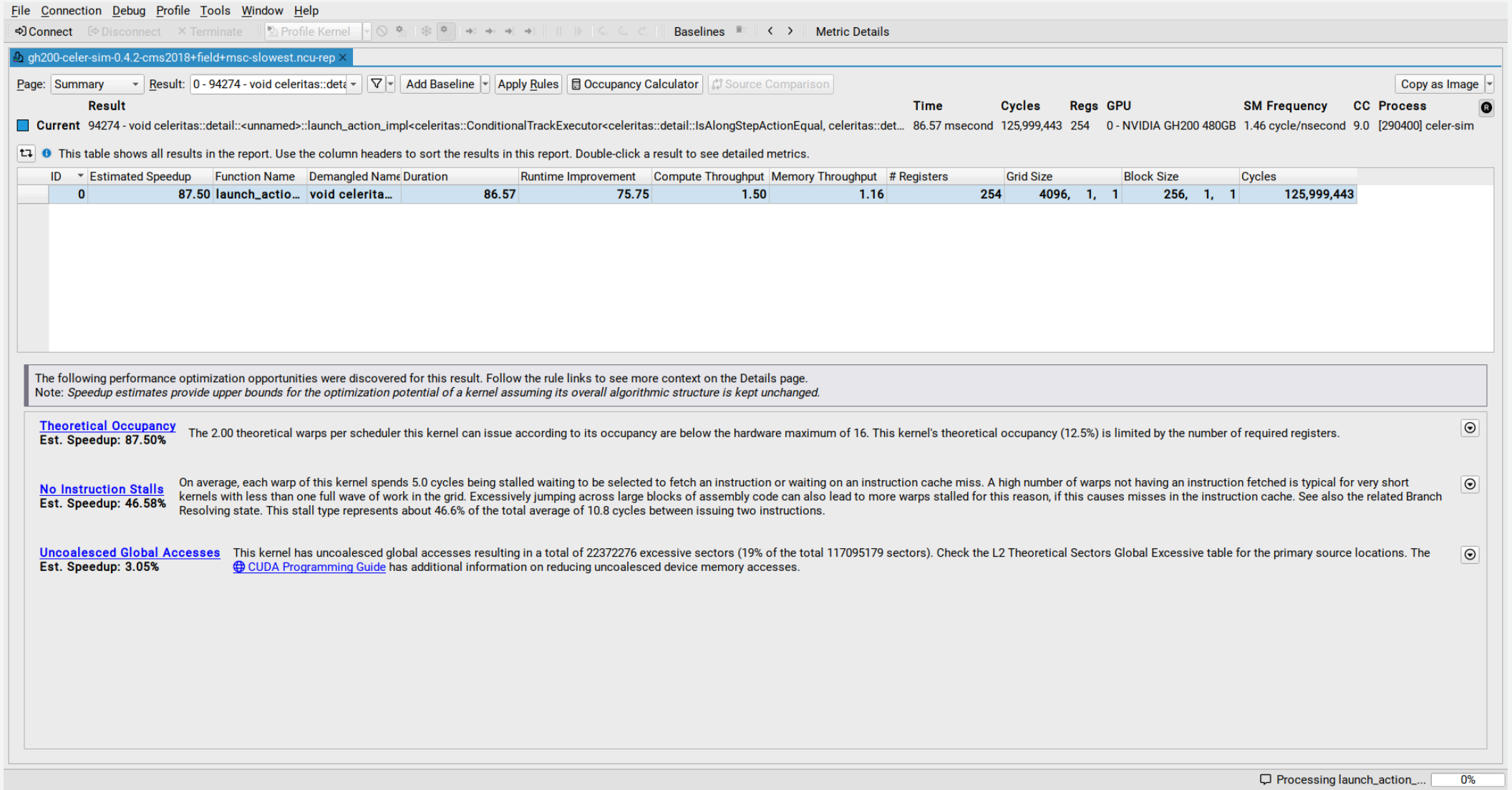
nsys: Host-Device Communication



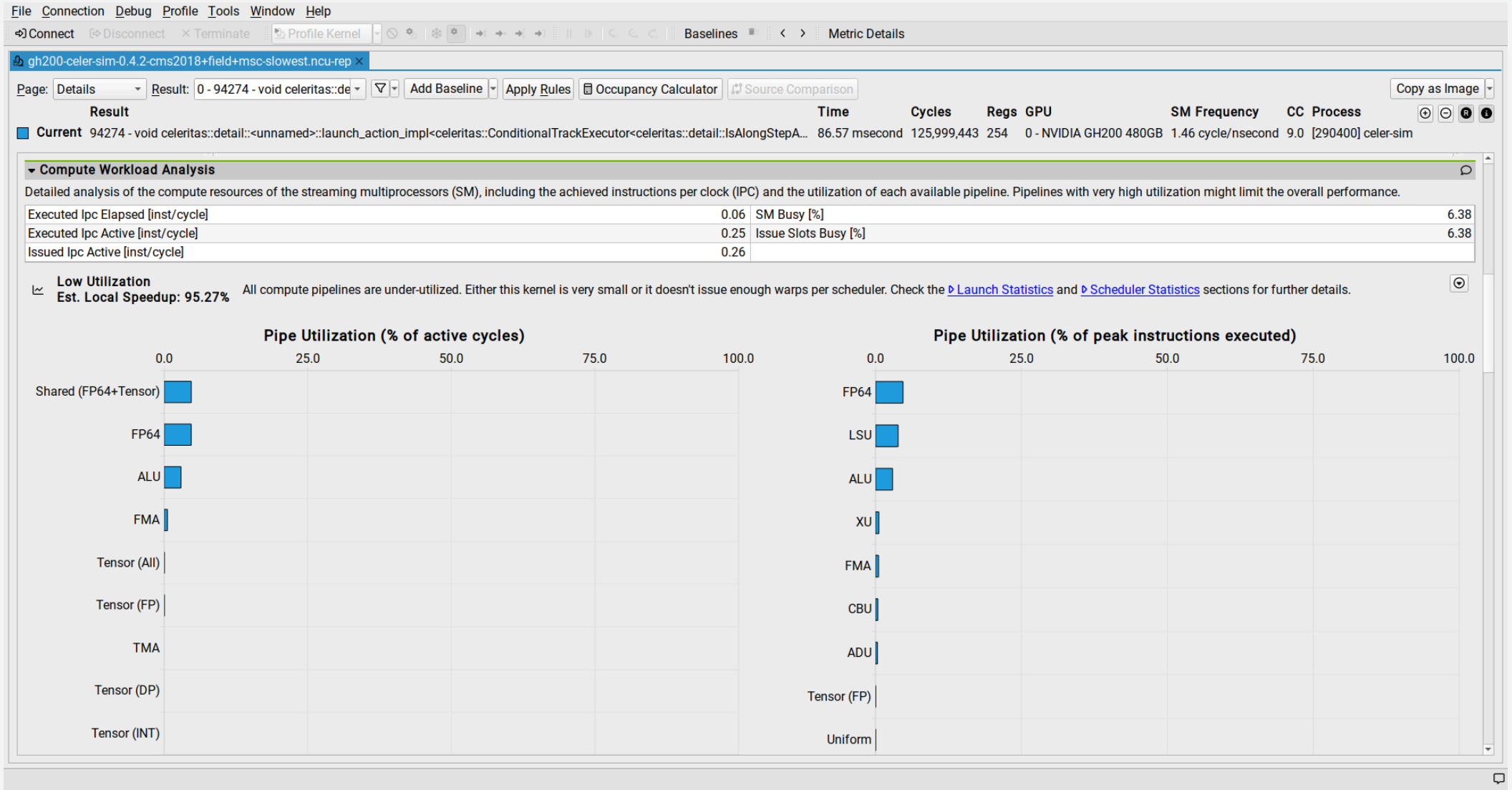
ncu: ERR_NVGPUCTRPERM

- Nvidia profiler counters require root or security mitigation disabling since 418.43 (2019-02-22). See ERR_NVGPUCTRPERM.

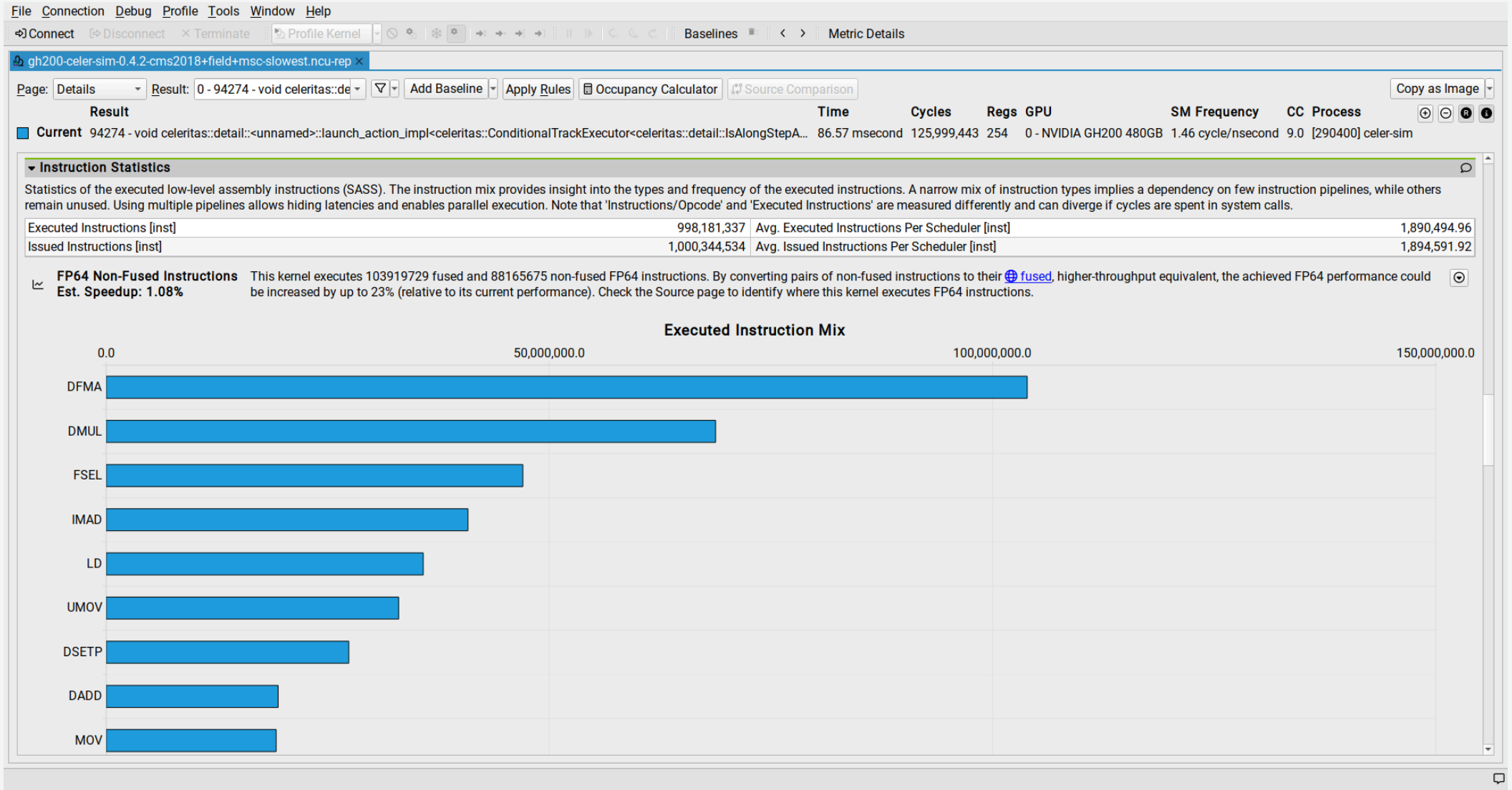
ncu: Summary along-step



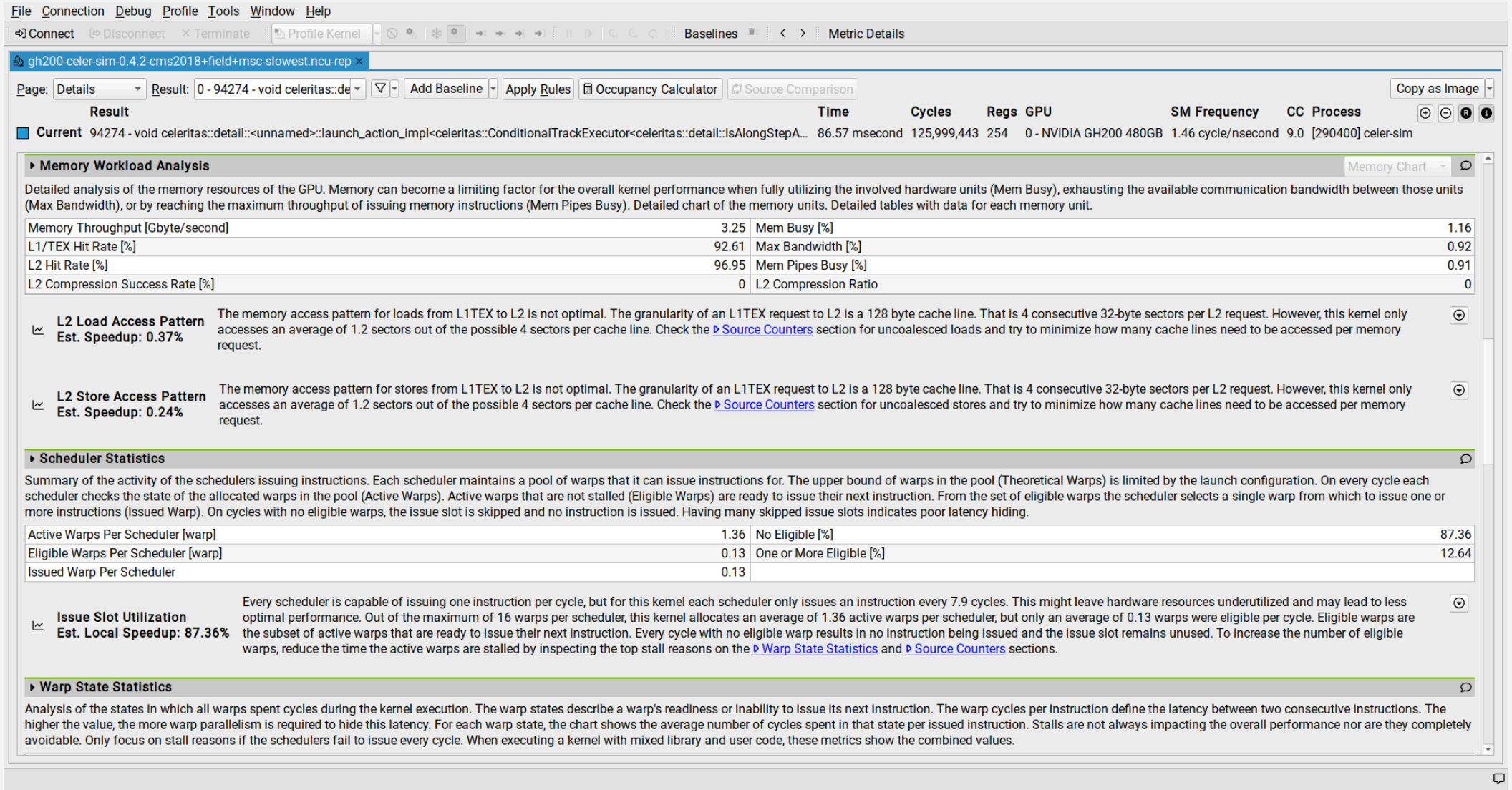
ncu: Compute


File Connection Debug Profile Tools Window Help
Connect Disconnect Terminate Profile Kernel Baselines Metric Details
gh200-celer-sim-0.4.2-cms2018+field+msc-slowest.ncu-rep
Page: Details Result: 0 - 94274 - void celeritas::de Add Baseline Apply Rules Occupancy Calculator Source Comparison Copy as Image
Result Time Cycles Regs GPU SM Frequency CC Process
Current 94274 - void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepA... 86.57 msecond 125,999,443 254 0 - NVIDIA GH200 480GB 1.46 cycle/nsecond 9.0 [290400] celer-sim
Compute Workload Analysis
Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.
Executed Ipc Elapsed [inst/cycle] 0.06
Executed Ipc Active [inst/cycle] 0.25
Issued Ipc Active [inst/cycle] 0.26
SM Busy [%] 6.38
Issue Slots Busy [%] 6.38
Low Utilization
Est. Local Speedup: 95.27%
All compute pipelines are under-utilized. Either this kernel is very small or it doesn't issue enough warps per scheduler. Check the Launch Statistics and Scheduler Statistics sections for further details.
Pipe Utilization (% of active cycles)
0.0 25.0 50.0 75.0 100.0
Shared (FP64+Tensor)
FP64
ALU
FMA
Tensor (All)
Tensor (FP)
TMA
Tensor (DP)
Tensor (INT)
Pipe Utilization (% of peak instructions executed)
0.0 25.0 50.0 75.0 100.0
FP64
LSU
ALU
XU
FMA
CBU
ADU
Tensor (FP)
Uniform

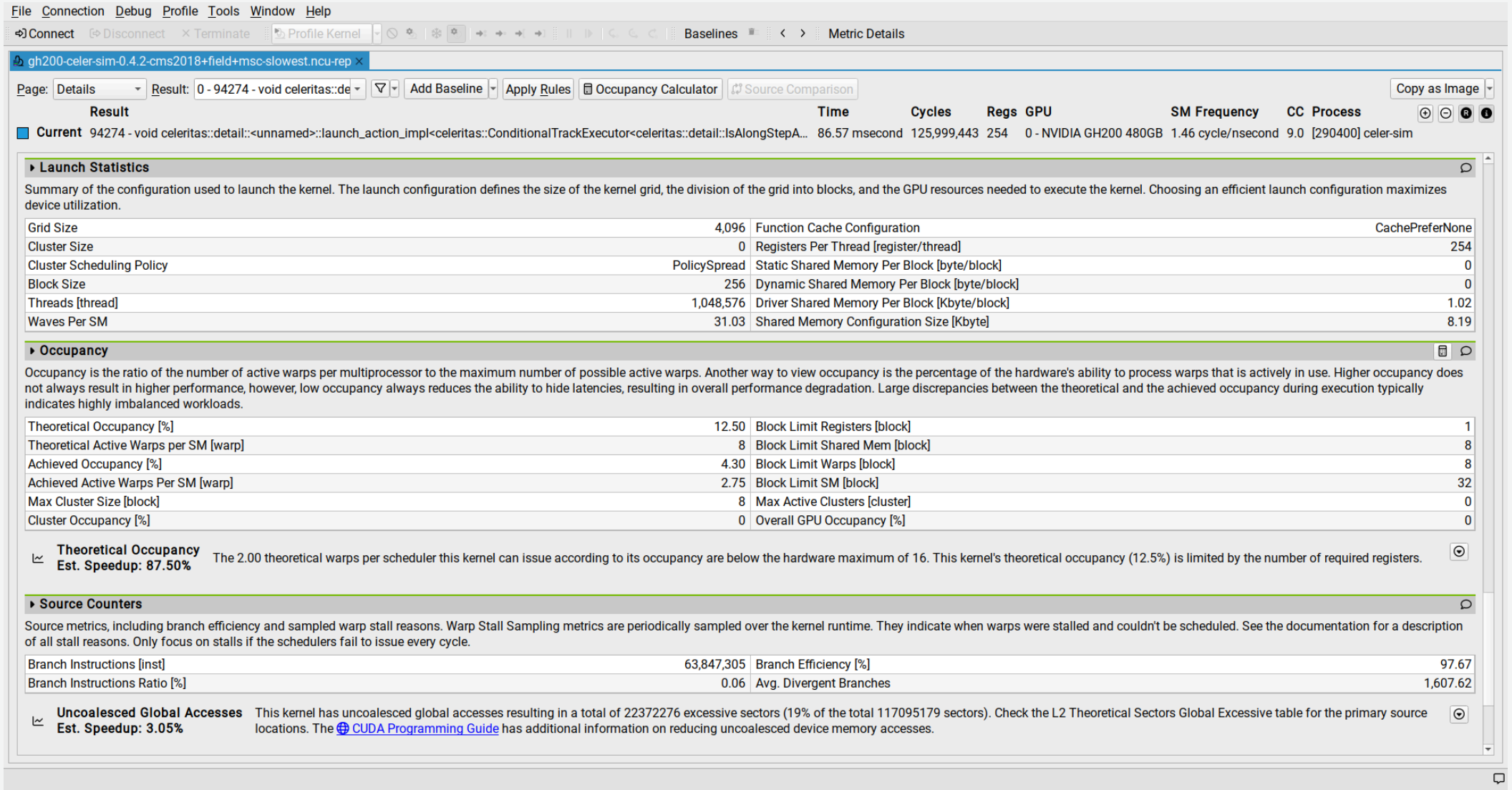
ncu: Instructions



ncu: Memory Access Pattern



ncu: Launch Statistics



File Connection Debug Profile Tools Window Help

Connect | Disconnect | Terminate | Profile Kernel | Baselines | Metric Details

gh200-celer-sim-0.4.2-cms2018+field+msc-slowest.ncu-rep

Page: Details | Result: 0 - 94274 - void celeritas::de | Add Baseline | Apply Rules | Occupancy Calculator | Source Comparison | Copy as Image

Result	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current 94274 - void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepA...	86.57 msecond	125,999,443	254	0 - NVIDIA GH200 480GB	1.46 cycle/nsecond	9.0	[290400] celer-sim

Launch Statistics

Summary of the configuration used to launch the kernel. The launch configuration defines the size of the kernel grid, the division of the grid into blocks, and the GPU resources needed to execute the kernel. Choosing an efficient launch configuration maximizes device utilization.

Grid Size	4,096	Function Cache Configuration	CachePreferNone
Cluster Size	0	Registers Per Thread [register/thread]	254
Cluster Scheduling Policy	PolicySpread	Static Shared Memory Per Block [byte/block]	0
Block Size	256	Dynamic Shared Memory Per Block [byte/block]	0
Threads [thread]	1,048,576	Driver Shared Memory Per Block [Kbyte/block]	1.02
Waves Per SM	31.03	Shared Memory Configuration Size [Kbyte]	8.19

Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Another way to view occupancy is the percentage of the hardware's ability to process warps that is actively in use. Higher occupancy does not always result in higher performance, however, low occupancy always reduces the ability to hide latencies, resulting in overall performance degradation. Large discrepancies between the theoretical and the achieved occupancy during execution typically indicates highly imbalanced workloads.

Theoretical Occupancy [%]	12.50	Block Limit Registers [block]	1
Theoretical Active Warps per SM [warp]	8	Block Limit Shared Mem [block]	8
Achieved Occupancy [%]	4.30	Block Limit Warps [block]	8
Achieved Active Warps Per SM [warp]	2.75	Block Limit SM [block]	32
Max Cluster Size [block]	8	Max Active Clusters [cluster]	0
Cluster Occupancy [%]	0	Overall GPU Occupancy [%]	0

Theoretical Occupancy
Est. Speedup: 87.50%
The 2.00 theoretical warps per scheduler this kernel can issue according to its occupancy are below the hardware maximum of 16. This kernel's theoretical occupancy (12.5%) is limited by the number of required registers.

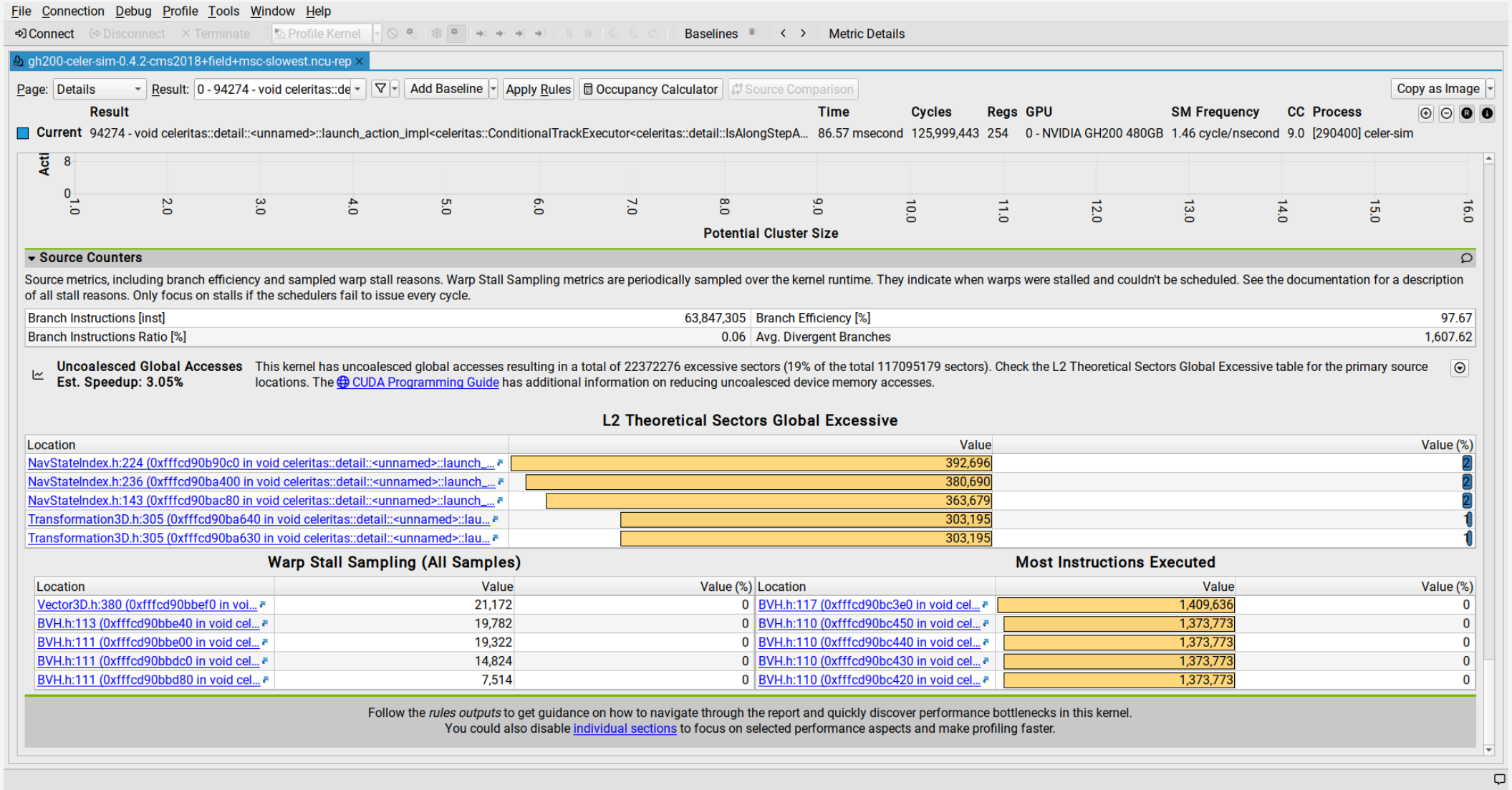
Source Counters

Source metrics, including branch efficiency and sampled warp stall reasons. Warp Stall Sampling metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be scheduled. See the documentation for a description of all stall reasons. Only focus on stalls if the schedulers fail to issue every cycle.

Branch Instructions [inst]	63,847,305	Branch Efficiency [%]	97.67
Branch Instructions Ratio [%]	0.06	Avg. Divergent Branches	1,607.62

Uncoalesced Global Accesses
Est. Speedup: 3.05%
This kernel has uncoalesced global accesses resulting in a total of 22372276 excessive sectors (19% of the total 117095179 sectors). Check the L2 Theoretical Sectors Global Excessive table for the primary source locations. The CUDA Programming Guide has additional information on reducing uncoalesced device memory accesses.

ncu: Source


File Connection Debug Profile Tools Window Help
Connect Disconnect Terminate Profile Kernel Baselines Metric Details
gh200-celer-sim-0.4.2-cms2018+field+msc-slowest.ncu-rep
Page: Details Result: 0 - 94274 - void celeritas::de Add Baseline Apply Rules Occupancy Calculator Source Comparison Copy as Image
Result Time Cycles Regs GPU SM Frequency CC Process
Current 94274 - void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepA... 86.57 msecond 125,999,443 254 0 - NVIDIA GH200 480GB 1.46 cycle/nsecond 9.0 [290400] celer-sim
Potential Cluster Size
Source Counters
Source metrics, including branch efficiency and sampled warp stall reasons. Warp Stall Sampling metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be scheduled. See the documentation for a description of all stall reasons. Only focus on stalls if the schedulers fail to issue every cycle.
Branch Instructions [inst] 63,847,305 Branch Efficiency [%] 97.67
Branch Instructions Ratio [%] 0.06 Avg. Divergent Branches 1,607.62
Uncoalesced Global Accesses
Est. Speedup: 3.05%
This kernel has uncoalesced global accesses resulting in a total of 22372276 excessive sectors (19% of the total 117095179 sectors). Check the L2 Theoretical Sectors Global Excessive table for the primary source locations. The CUDA Programming Guide has additional information on reducing uncoalesced device memory accesses.
L2 Theoretical Sectors Global Excessive
Location Value Value (%)
NavStateIndex.h:224 (0xfffcd90b90c0 in void celeritas::detail::<unnamed>::launch_... 392,696 2
NavStateIndex.h:236 (0xfffcd90ba400 in void celeritas::detail::<unnamed>::launch_... 380,690 2
NavStateIndex.h:143 (0xfffcd90bac80 in void celeritas::detail::<unnamed>::launch_... 363,679 2
Transformation3D.h:305 (0xfffcd90ba640 in void celeritas::detail::<unnamed>::lau... 303,195 1
Transformation3D.h:305 (0xfffcd90ba630 in void celeritas::detail::<unnamed>::lau... 303,195 1
Warp Stall Sampling (All Samples)
Location Value Value (%)
Vector3D.h:380 (0xfffcd90bbef0 in voi... 21,172 0
BVH.h:113 (0xfffcd90bbe40 in void cel... 19,782 0
BVH.h:111 (0xfffcd90bbe00 in void cel... 19,322 0
BVH.h:111 (0xfffcd90bbdc0 in void cel... 14,824 0
BVH.h:111 (0xfffcd90bbd80 in void cel... 7,514 0
Most Instructions Executed
Location Value Value (%)
BVH.h:117 (0xfffcd90bc3e0 in void cel... 1,409,636 0
BVH.h:110 (0xfffcd90bc450 in void cel... 1,373,773 0
BVH.h:110 (0xfffcd90bc440 in void cel... 1,373,773 0
BVH.h:110 (0xfffcd90bc430 in void cel... 1,373,773 0
BVH.h:110 (0xfffcd90bc420 in void cel... 1,373,773 0
Follow the rules outputs to get guidance on how to navigate through the report and quickly discover performance bottlenecks in this kernel.
You could also disable individual sections to focus on selected performance aspects and make profiling faster.