



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**

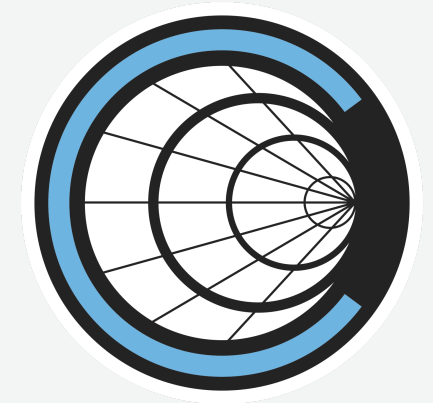
Profile

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)

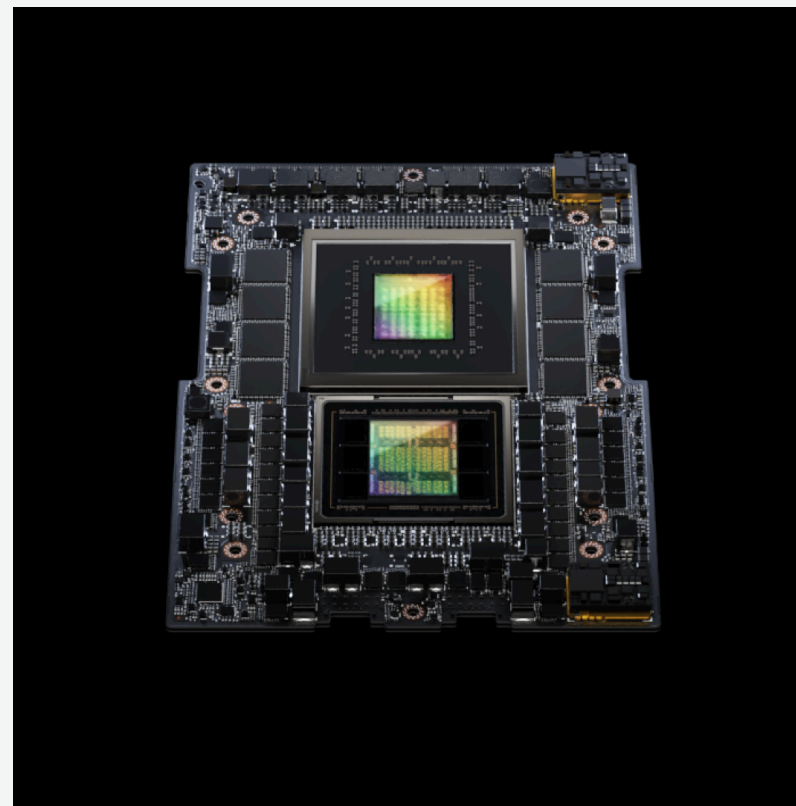
- Highly-parallel many-core co-processors
- Optimised for throughput
- (Relatively) Low volume of High-bandwidth memory
- Power efficient (for suitable workloads)
- Often connected via low-bandwidth PCIe



Titan Xp & Titan V GPUs

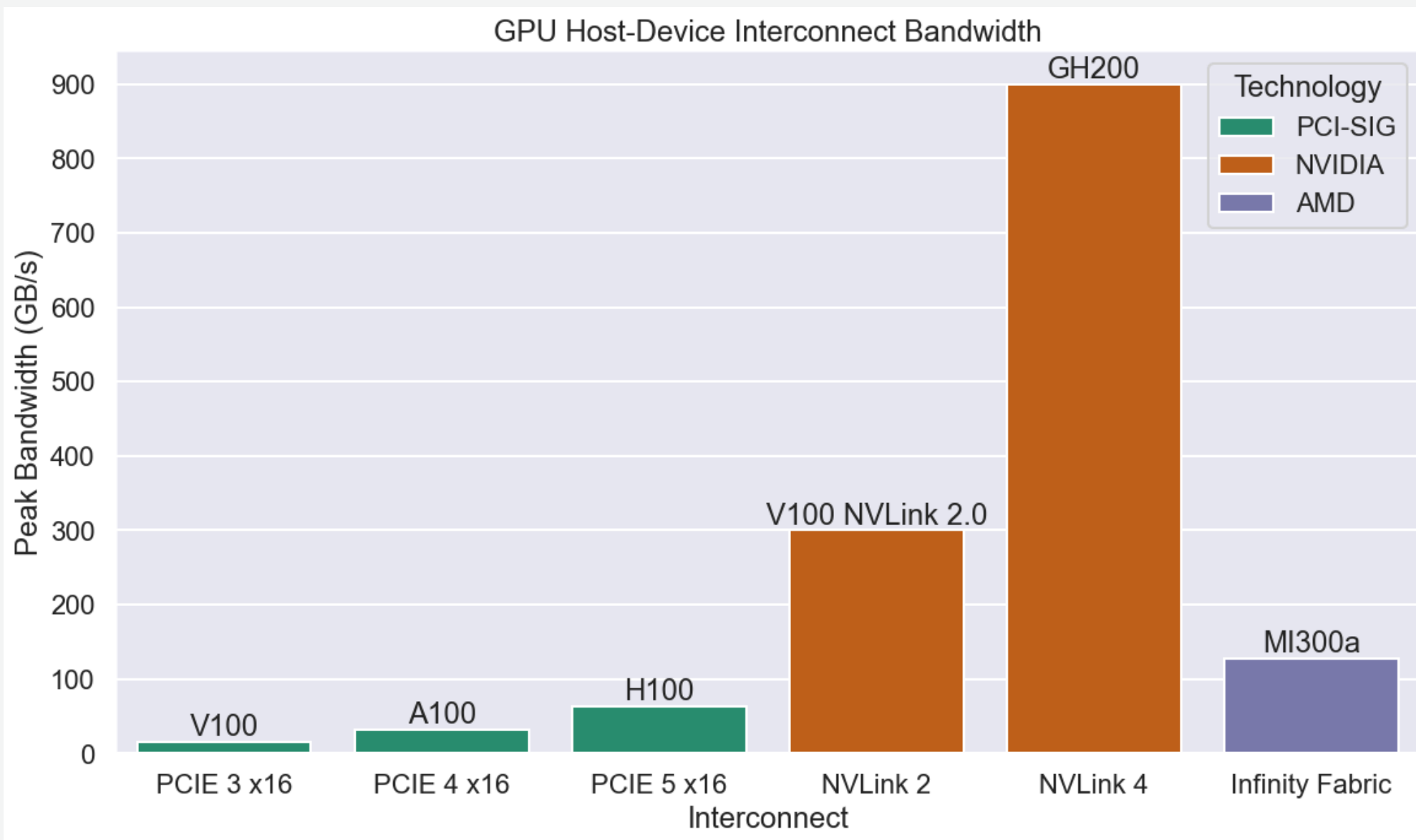
NVIDIA Grace Hopper Superchip

- 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

```

1 $ ctest
2 # ...
3 99% tests passed, 2 tests failed out of 203

```

```

1 $ ctest --rerun-failed --output-on-failure
2 # ...
3 1/2 Test #158: celeritas/mat/Material .....***Failed
4     Error regular expression found in output. Regex=[tests FAILED] 0.
5 # ...
6 2/2 Test #160: celeritas/phys/Particle ..... Passed    0.61 sec
7
8 50% tests passed, 1 tests failed out of 2

```

JSON Comparison	<code>mass_radiation_coeff</code>
-----------------	-----------------------------------

Expected	0.03605392839455309
----------	---------------------

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)

```

1 $ time ./bin/celer-sim cms2018+field+msc
2 # ...
3 real    1m13.997s
4 user    1m7.096s
5 sys     0m0.871s

```

```

1 {
2     "_exe": "celer-sim",
3     "_format": "celer-sim",
4     "_geometry": "vecgeom",
5     "_instance": 0,
6     "_name": [
7         "cms2018+field+msc",
8         "vecgeom",
9         "gpu"
10    ],
11    "_outdir": "cms2018+field+msc-vecgeom-gpu",
12    "_timeout": 600.0,
13    "_use_celeritas": true,
14    "_version": "0.4.2",
15    "action_diagnostic": false,
16    "brem_combined": false,
17    "cuda_heap_size": null,
18    "cuda_stack_size": 8192,
19    "default_stream": false,
20    "environ": {},
21    "event_file": null,
22    "field": [
23        0.0,
24        0.0,
25        1.0
26    ],
27    "field_options": {
28        "delta_chord": 0.025,
29        "delta_intersection": 1e-05,
30        "epsilon_rel_max": 0.001,
31        "epsilon_step": 1e-05,
32        "errcon": 0.0001,

```

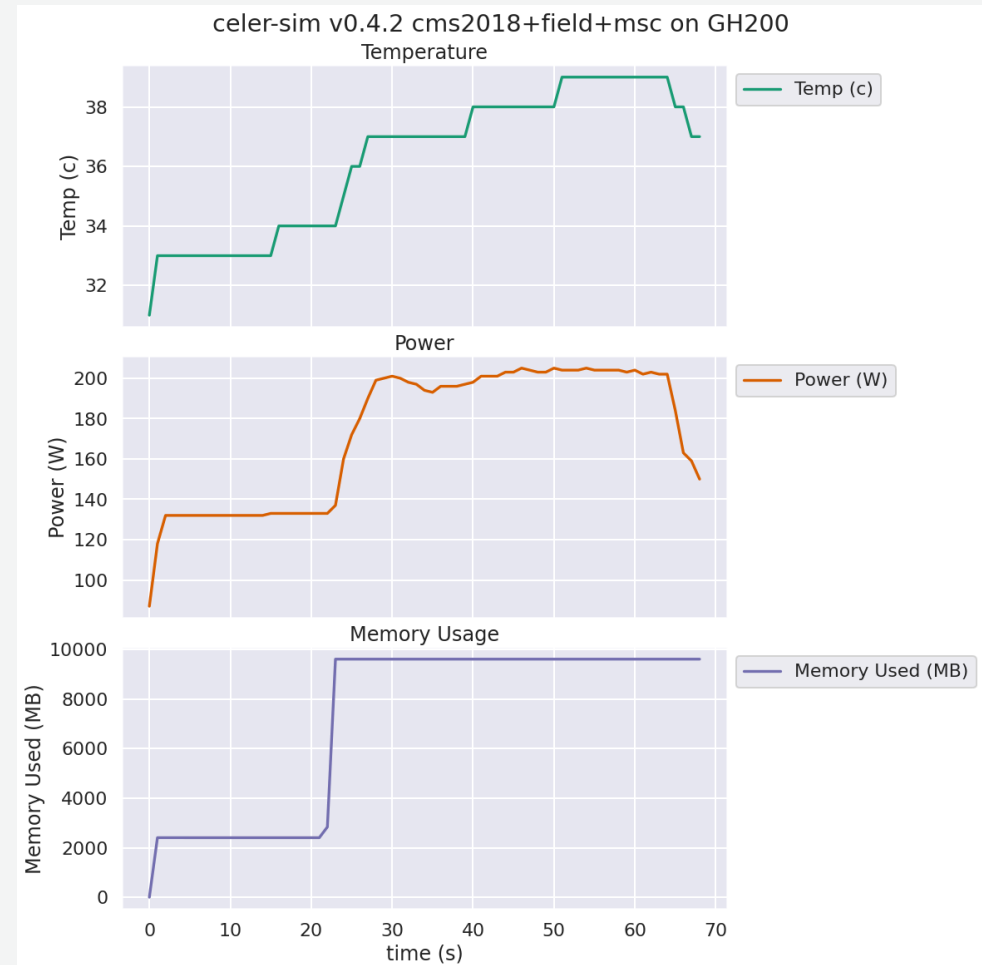
Power Consumption Monitoring

- Can use `nvm1` 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

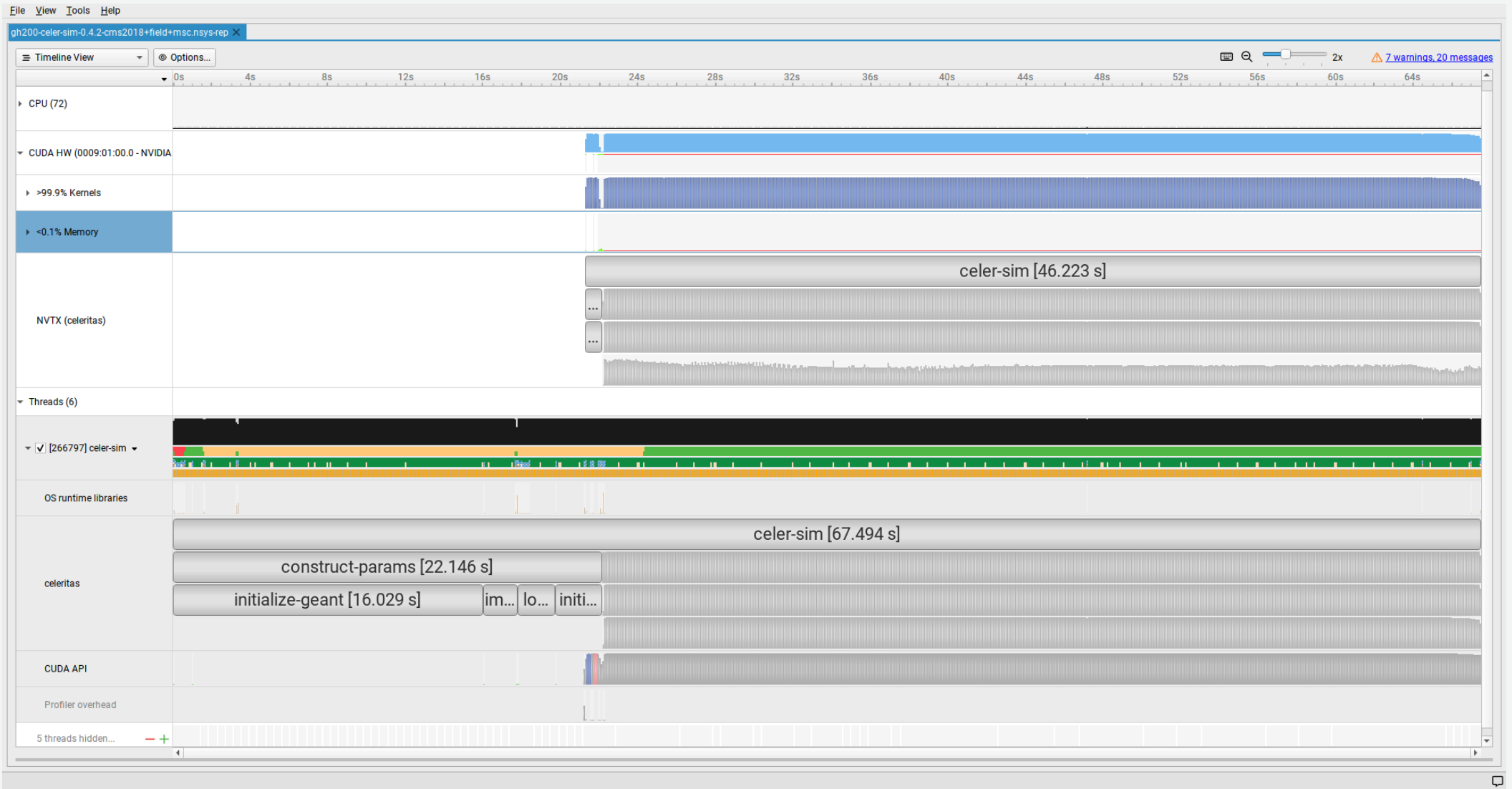
- System-wide performance analysis
- CPU + GPU
- Visualise a timeline of events
- CUDA API information, kernel block sizes, etc
- Pascal GPUs or newer (SM 60+)



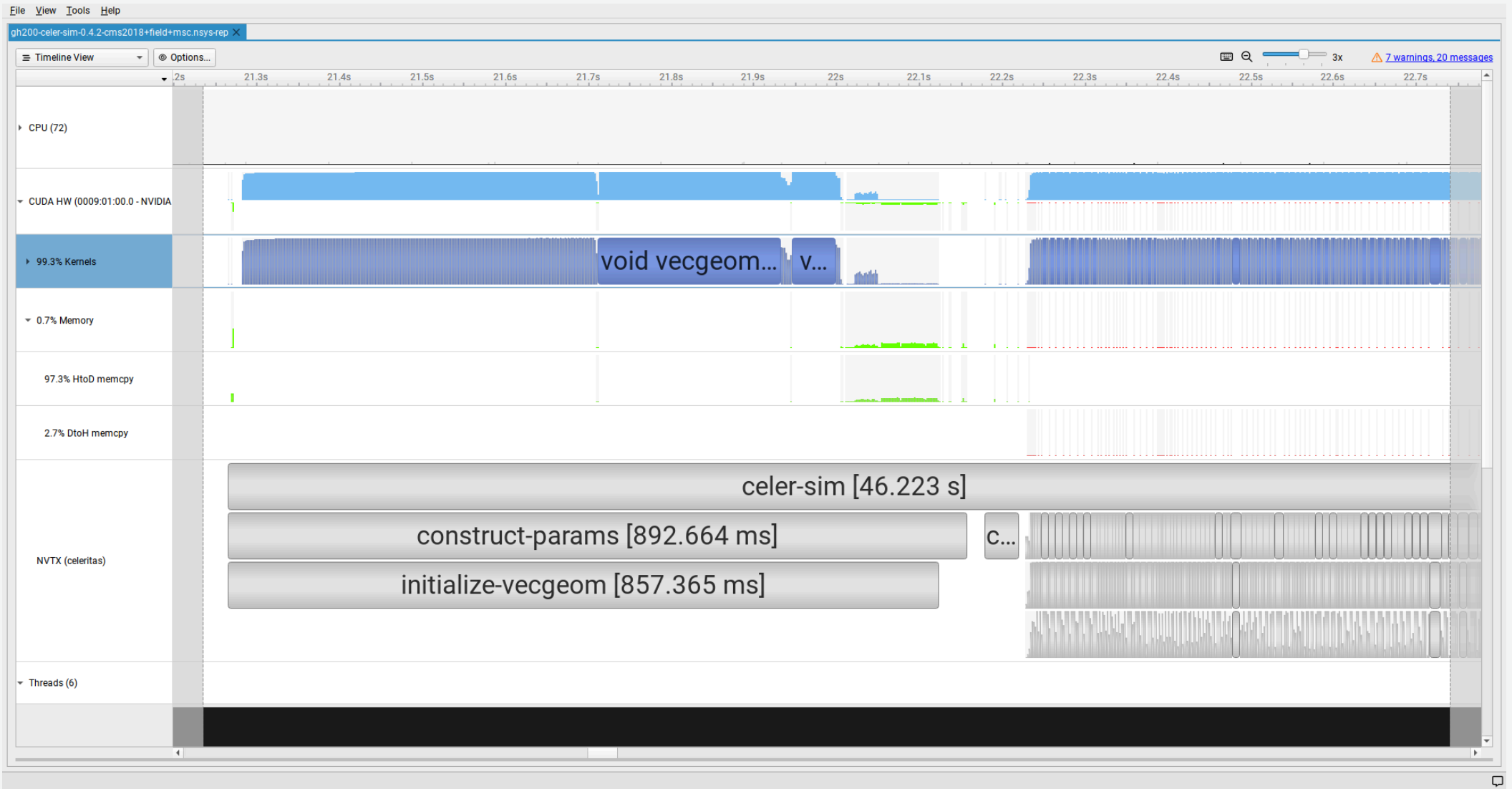
```
1 nsys profile -o timeline ./bin/celer-sim input.json
2 nsys-ui timeline.nsys-rep
```

- Enable NVTX in Celeritas via `CELER_ENABLE_PROFILING=1`

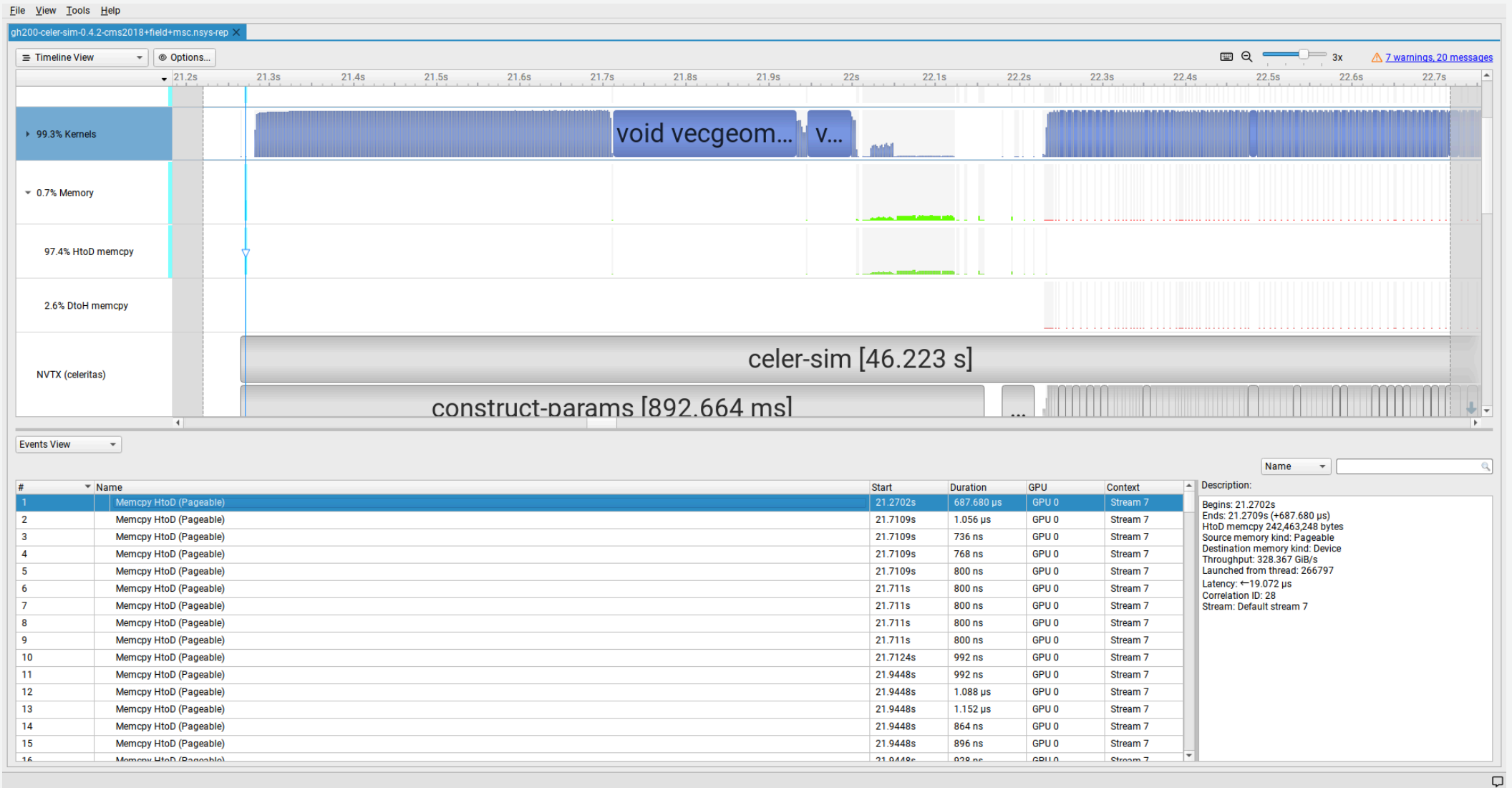
nsys: Timeline



nsys: Host-Device Communication



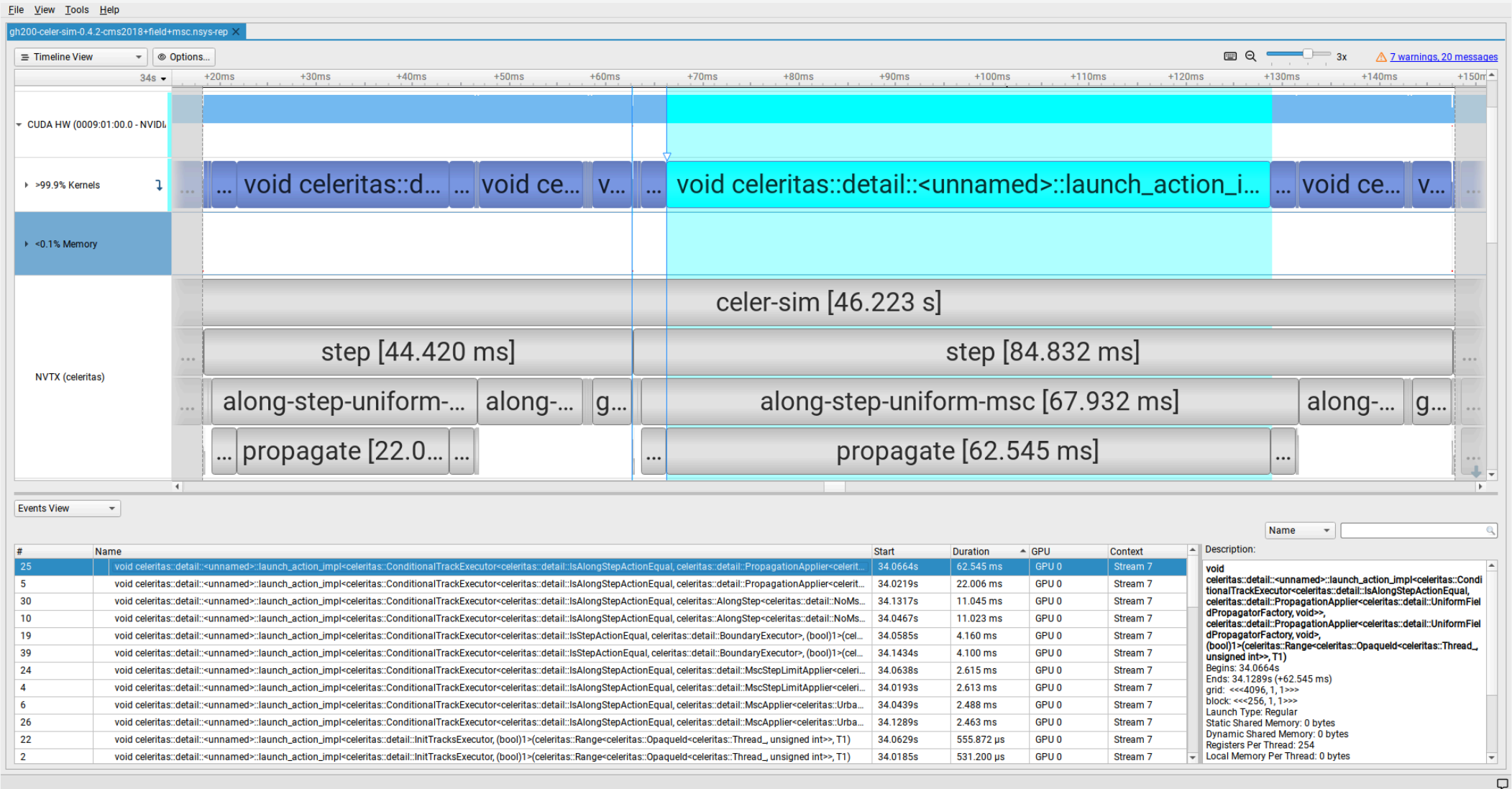
nsys: Host-Device Communication



242MB, 690μs @ 328GB/s

Profiling Celeritas - GridPP51 & SWIFT-HEP07

nsys: Longest Duration Kernel



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 \geq 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 x

Page: Summary Result: 0 - 148030 - void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsStepActionEqual, celeritas::detail::Bou... Time Cycles Regs GPU SM Frequency CC Process

2.65 msecond 4,056,724 214 0 - NVIDIA GH200 480GB 1.53 cycle/nsecond 9.0 [273958] celer-sim

Copy as Image

Result

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	Cy
0	87.50	launch_actio...	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsStepActionEqual, celeritas::detail::Bou...	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<celeritas::ConditionalTrackExecutor<celeritas::detail::IsStepActionEqual, celeritas::detail::Bou...	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<celeritas::ConditionalTrackExecutor<celeritas::detail::IsStepActionEqual, celeritas::detail::Bou...	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<celeritas::ConditionalTrackExecutor<celeritas::detail::IsStepActionEqual, celeritas::detail::Bou...	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<celeritas::ConditionalTrackExecutor<celeritas::detail::IsStepActionEqual, celeritas::detail::Bou...	0.37	0.19	44.80	40.05	62	4096, 1, ...	256, 1, ...	
7	87.50	launch_actio...	void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsStepActionEqual, celeritas::detail::Bou...	1.11	0.97	9.73	10.65	214	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

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.

GPU Throughput

Speed Of Light (SOL) [%]

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

File Connection Debug Profile Tools Window Help

Connect Disconnect Terminate Profile Kernel Baselines Metric Details

gh200-celer-sim-0.4.2-cms2018+field+mso-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

Scheduler Statistics

Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

Active Warps Per Scheduler [warp]	1.36	No Eligible [%]	87.36
Eligible Warps Per Scheduler [warp]	0.13	One or More Eligible [%]	12.64
Issued Warp Per Scheduler	0.13		

Issue Slot Utilization
Est. Local Speedup: 87.36%

Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 7.9 cycles. This might leave hardware resources underutilized and may lead to less optimal performance. Out of the maximum of 16 warps per scheduler, this kernel allocates an average of 1.36 active warps per scheduler, but only an average of 0.13 warps were eligible per cycle. Eligible warps are the subset of active warps that are ready to issue their next instruction. Every cycle with no eligible warp results in no instruction being issued and the issue slot remains unused. To increase the number of eligible warps, reduce the time the active warps are stalled by inspecting the top stall reasons on the [Warp State Statistics](#) and [Source Counters](#) sections.

Warps Per Scheduler

Category	Value
GPU Maximum Warps Per Scheduler	16.0
Theoretical Warps Per Scheduler	16.0
Active Warps Per Scheduler	1.36
Eligible Warps Per Scheduler	0.13
Issued Warp Per Scheduler	0.13

Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.

Warp Cycles Per Issued Instruction [cycle]	10.78	Avg. Active Threads Per Warp	1.63
--	-------	------------------------------	------

ncu: Warp state

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 x

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

Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.

Warp Cycles Per Issued Instruction [cycle]	10.78	Avg. Active Threads Per Warp	1.63
Warp Cycles Per Executed Instruction [cycle]	10.80	Avg. Not Predicated Off Threads Per Warp	1.59

No Instruction Stalls **Est. Speedup: 46.58%** On average, each warp of this kernel spends 5.0 cycles being stalled waiting to be selected to fetch an instruction or waiting on an instruction cache miss. A high number of warps not having an instruction fetched is typical for very short kernels with less than one full wave of work in the grid. Excessively jumping across large blocks of assembly code can also lead to more warps stalled for this reason, if this causes misses in the instruction cache. See also the related Branch Resolving state. This stall type represents about 46.6% of the total average of 10.8 cycles between issuing two instructions.

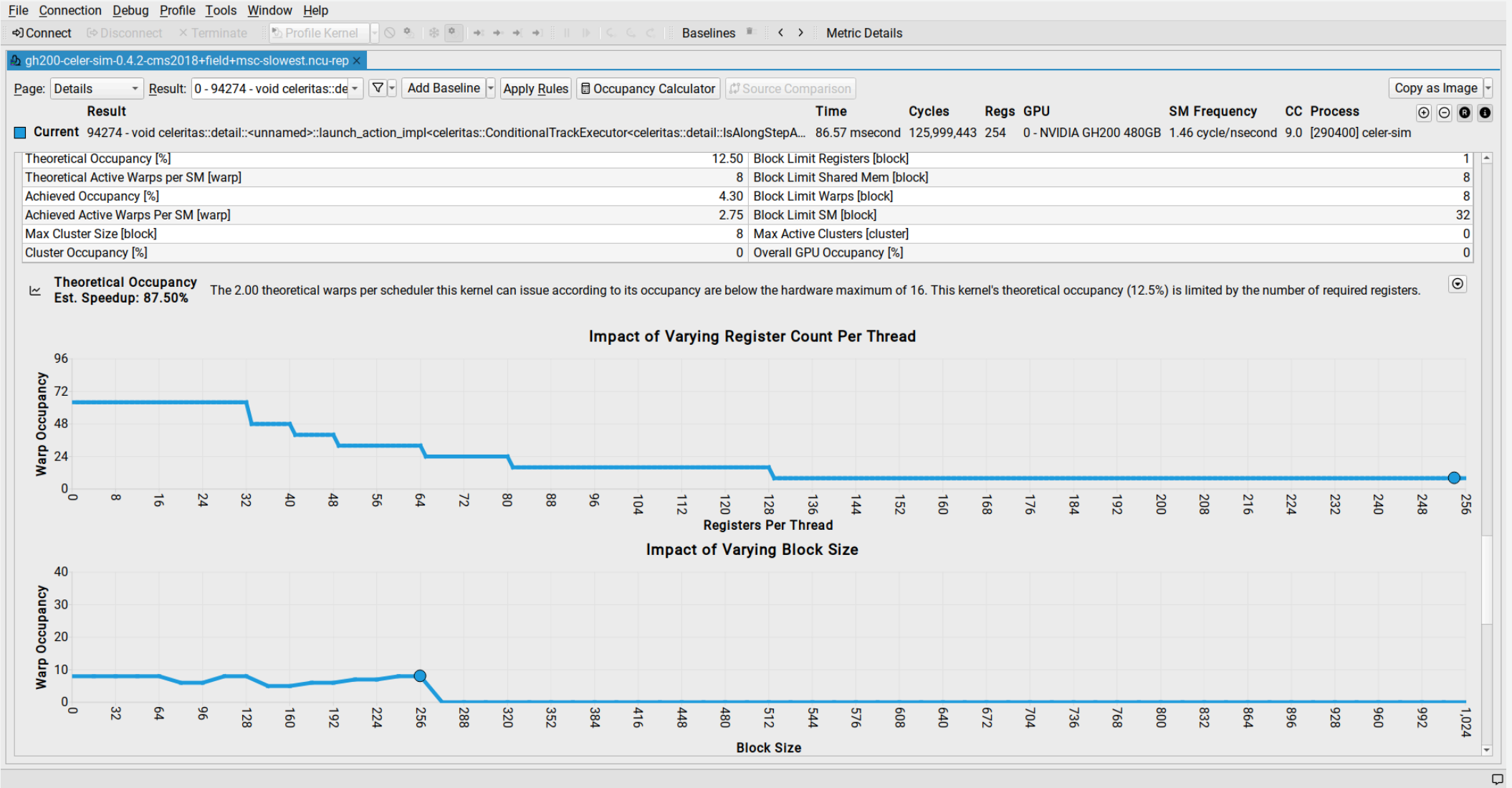
Warp Stall Check the [Warp Stall Sampling \(All Samples\)](#) table for the top stall locations in your source based on sampling data. The [Kernel Profiling Guide](#) provides more details on each stall reason.

Thread Divergence **Est. Speedup: 1.43%** Instructions are executed in warps, which are groups of 32 threads. Optimal instruction throughput is achieved if all 32 threads of a warp execute the same instruction. The chosen launch configuration, early thread completion, and divergent flow control can significantly lower the number of active threads in a warp per cycle. This kernel achieves an average of 1.6 threads being active per cycle. This is further reduced to 1.6 threads per warp due to predication. The compiler may use predication to avoid an actual branch. Instead, all instructions are scheduled, but a per-thread condition code or predicate controls which threads execute the instructions. Try to avoid different execution paths within a warp when possible.

Warp State (All Cycles)

Warp State	Approximate Cycles
Stall No Instruction	5.0
Stall Wait	4.0
Stall Long Scoreboard	3.5
Selected	2.5
Stall Branch Resolving	1.5

ncu: Occupancy



ncu: Performance Monitor Sampling

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 x

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

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.

0s ms +5ms +10ms +15ms +20ms +25ms +30ms +35ms +40ms +45ms +50ms +55ms +60ms +65ms +70ms +75ms +80ms

- Overview
 - Blocks Launched
 - SM Active Cycles
 - Executed Ipc Active
- SM
 - SM Throughput
 - SM ALU Pipe Throughput
 - SM Tensor Pipe Throughput
- DRAM
 - DRAM Throughput
 - DRAM Read Throughput
 - DRAM Write Throughput
- L1 Cache
 - Hit Rate
 - Wavefronts (Data)

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.

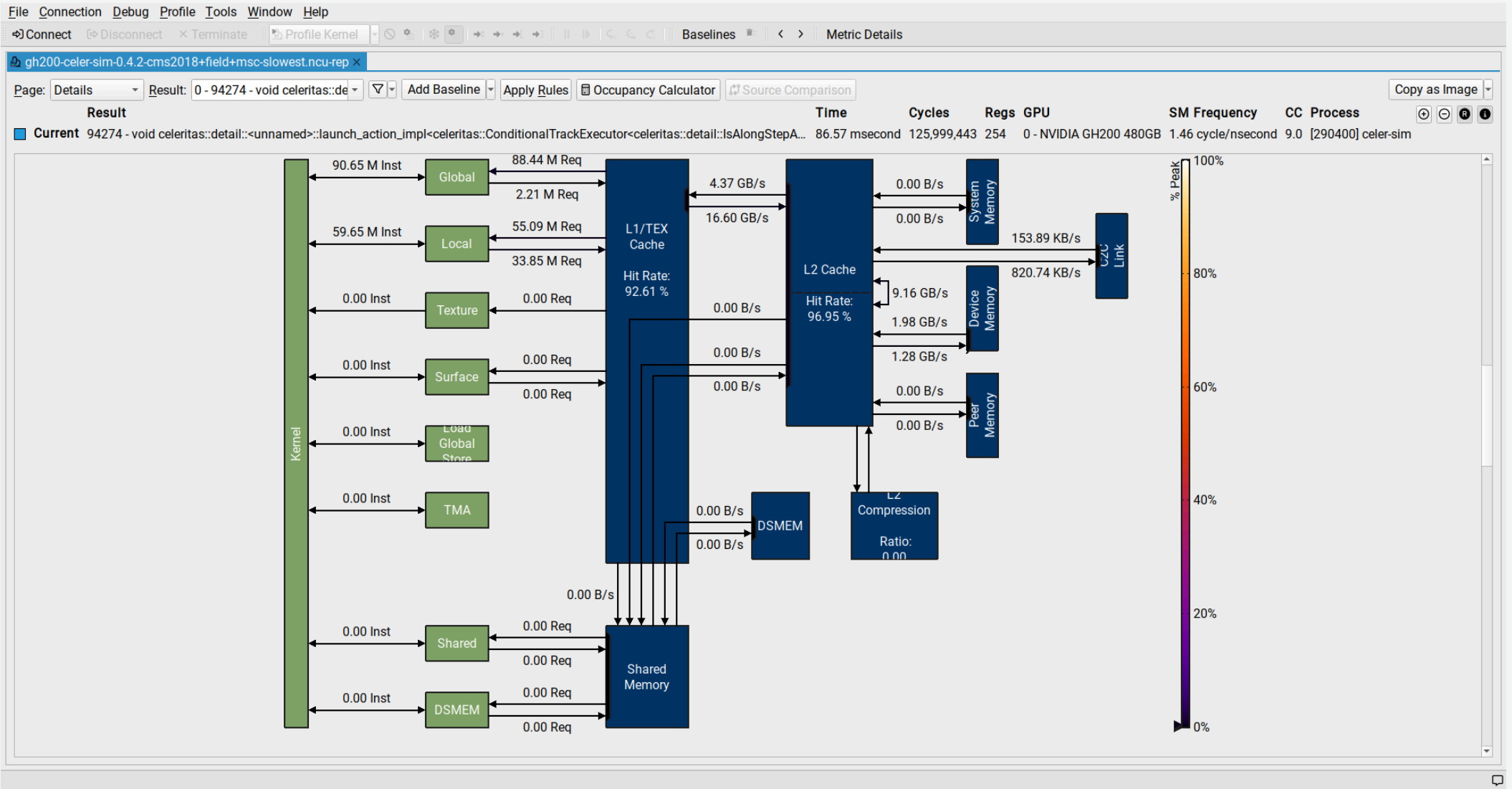
Executed Ipc Elapsed [inst/cycle]	0.06	SM Busy [%]	6.38
Executed Ipc Active [inst/cycle]	0.25	Issue Slots Busy [%]	6.38
Issued Ipc Active [inst/cycle]	0.26		

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.

Memory Workload Analysis Memory Chart

- Nsight Compute \geq 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`

```
1 include/VecGeom/base/Transformation3D.h: In member function
2   'vecgeom::cxx::Vector3D<double>
3   vecgeom::cxx::Transformation3D::Translation() const':
4 include/VecGeom/base/Transformation3D.h:213:3: note: parameter
5   passing for argument of type 'vecgeom::cxx::Vector3D<double>'
6   when C++17 is enabled changed to match C++14 in GCC 10.1
7   213 |   {
8       |   ^
```


Celeritas test suite on GH200

```
1 $ ctest
2 # ...
3 99% tests passed, 2 tests failed out of 203
4
5 Label Time Summary:
6 app           = 108.80 sec*proc (11 tests)
7 gpu           = 101.33 sec*proc (43 tests)
8 nomemcheck    = 107.88 sec*proc (9 tests)
9 unit          =  33.99 sec*proc (191 tests)
10
11 Total Test time (real) = 140.78 sec
12
13 The following tests FAILED:
14     158 - celeritas/mat/Material (Failed)
15     160 - celeritas/phys/Particle (SEGFAULT)
```

Celeritas test suite on GH200

```

1 $ ctest --rerun-failed --output-on-failure
2 # ...
3 1/2 Test #158: celeritas/mat/Material .....***Failed
4     Error regular expression found in output. Regex=[tests FAILED] 0.
5 # ...
6 2/2 Test #160: celeritas/phys/Particle ..... Passed    0.61 sec
7
8 50% tests passed, 1 tests failed out of 2
9
10 The following tests FAILED:
11     158 - celeritas/mat/Material (Failed)

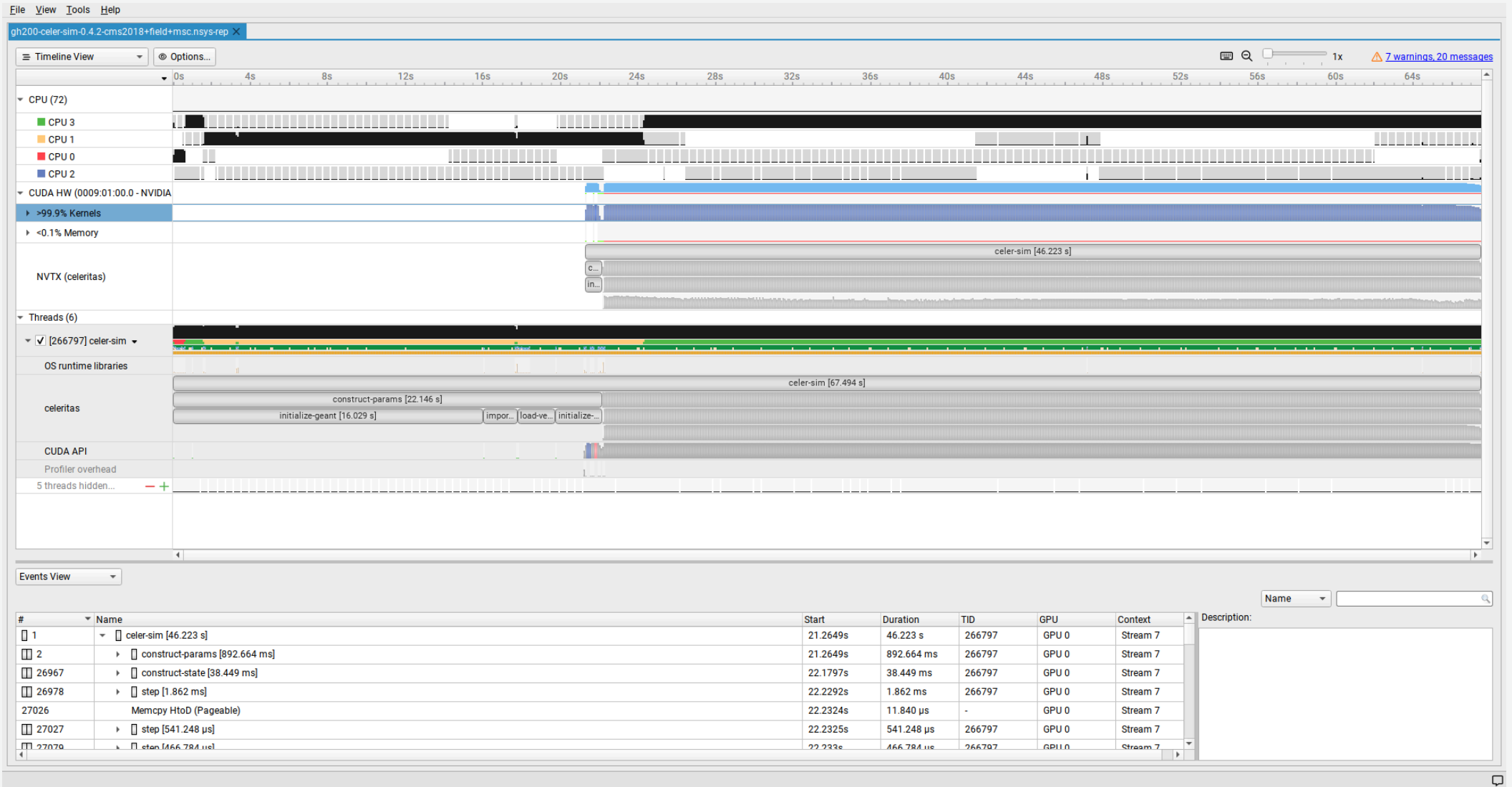
```

JSON Comparison	<code>mass_radiation_coeff</code>
Expected	0.03605392839455309
Actual	0.0360539283945531

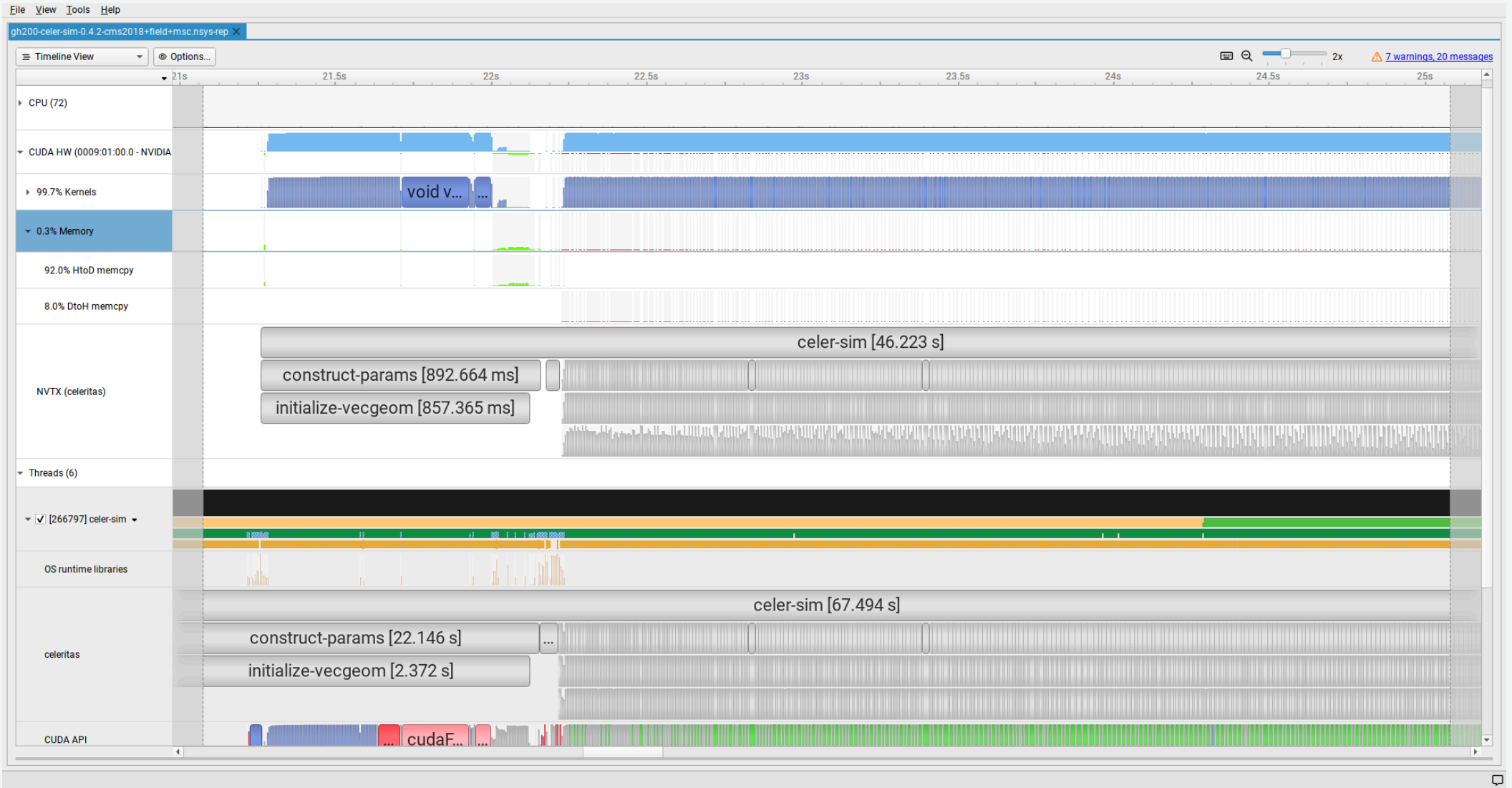
Running the scenario

```
1 $ time ./bin/celer-sim cms2018+field+msc.json
2 status: Loading input and initializing problem data
3 status: Initializing Geant4 run manager
4 status: Initializing Geant4 geometry
5 info: Loading Geant4 geometry from GDML at /path/to/cms2018.gdml
6 status: Building Geant4 physics tables
7 status: Transferring data from Geant4
8 status: Loading external elemental data
9 status: Loading VecGeom geometry from GDML at /path/to/cms2018.gdml
10 status: Initializing tracking information
11 celeritas/src/celeritas/geo/GeoMaterialParams.cc:205: warning: Some ge
12
13 # ...
14
15 real    1m13.997s
16 user    1m7.096s
17 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

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: Summary Result: 0 - 94274 - void celeritas::detail::<unnamed>::launch_action_impl<celeritas::ConditionalTrackExecutor<celeritas::detail::IsAlongStepActionEqual, celeritas::det... 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::IsAlongStepActionEqual, celeritas::det...	86.57 msecond	125,999,443	254	0 - NVIDIA GH200 480GB	1.46 cycle/nsecond	9.0	[290400] 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	Compute Throughput	Memory Throughput	# Registers	Grid Size	Block Size	Cycles
0	87.50	launch_actio...	void celerita...	86.57	75.75	1.50	1.16	254	4096, 1, 1	256, 1, 1	125,999,443

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.
- No Instruction Stalls** Est. Speedup: 46.58% On average, each warp of this kernel spends 5.0 cycles being stalled waiting to be selected to fetch an instruction or waiting on an instruction cache miss. A high number of warps not having an instruction fetched is typical for very short kernels with less than one full wave of work in the grid. Excessively jumping across large blocks of assembly code can also lead to more warps stalled for this reason, if this causes misses in the instruction cache. See also the related Branch Resolving state. This stall type represents about 46.6% of the total average of 10.8 cycles between issuing two instructions.
- 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.

Processing launch_action... 0%

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

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)

Pipeline Type	Utilization (%)
Shared (FP64+Tensor)	~5
FP64	~5
ALU	~5
FMA	~2
Tensor (All)	~0
Tensor (FP)	~0
TMA	~0
Tensor (DP)	~0
Tensor (INT)	~0

Pipe Utilization (% of peak instructions executed)

Pipeline Type	Utilization (%)
FP64	~5
LSU	~5
ALU	~5
XU	~2
FMA	~2
CBU	~1
ADU	~1
Tensor (FP)	~0
Uniform	~0

ncu: Instructions

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 x

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

Instruction Statistics

Statistics of the executed low-level assembly instructions (SASS). The instruction mix provides insight into the types and frequency of the executed instructions. A narrow mix of instruction types implies a dependency on few instruction pipelines, while others remain unused. Using multiple pipelines allows hiding latencies and enables parallel execution. Note that 'Instructions/Opcode' and 'Executed Instructions' are measured differently and can diverge if cycles are spent in system calls.

Executed Instructions [inst]	998,181,337	Avg. Executed Instructions Per Scheduler [inst]	1,890,494.96
Issued Instructions [inst]	1,000,344,534	Avg. Issued Instructions Per Scheduler [inst]	1,894,591.92

FP64 Non-Fused Instructions This kernel executes 103919729 fused and 88165675 non-fused FP64 instructions. By converting pairs of non-fused instructions to their [fused](#), higher-throughput equivalent, the achieved FP64 performance could be increased by up to 23% (relative to its current performance). Check the Source page to identify where this kernel executes FP64 instructions.

Est. Speedup: 1.08%

Executed Instruction Mix

Instruction Type	Approximate Count
DFMA	125,000,000
DMUL	75,000,000
FSEL	50,000,000
IMAD	45,000,000
LD	40,000,000
UMOV	35,000,000
DSETP	30,000,000
DADD	20,000,000
MOV	20,000,000

ncu: Memory Access Pattern

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

Memory Workload Analysis

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.

Memory Throughput [Gbyte/second]	3.25	Mem Busy [%]	1.16
L1/TEX Hit Rate [%]	92.61	Max Bandwidth [%]	0.92
L2 Hit Rate [%]	96.95	Mem Pipes Busy [%]	0.91
L2 Compression Success Rate [%]	0	L2 Compression Ratio	0

L2 Load Access Pattern
Est. Speedup: 0.37%

The memory access pattern for loads from L1TEX to L2 is not optimal. The granularity of an L1TEX request to L2 is a 128 byte cache line. That is 4 consecutive 32-byte sectors per L2 request. However, this kernel only accesses an average of 1.2 sectors out of the possible 4 sectors per cache line. Check the [Source Counters](#) section for uncoalesced loads and try to minimize how many cache lines need to be accessed per memory request.

L2 Store Access Pattern
Est. Speedup: 0.24%

The memory access pattern for stores from L1TEX to L2 is not optimal. The granularity of an L1TEX request to L2 is a 128 byte cache line. That is 4 consecutive 32-byte sectors per L2 request. However, this kernel only accesses an average of 1.2 sectors out of the possible 4 sectors per cache line. Check the [Source Counters](#) section for uncoalesced stores and try to minimize how many cache lines need to be accessed per memory request.

Scheduler Statistics

Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

Active Warps Per Scheduler [warp]	1.36	No Eligible [%]	87.36
Eligible Warps Per Scheduler [warp]	0.13	One or More Eligible [%]	12.64
Issued Warp Per Scheduler	0.13		

Issue Slot Utilization
Est. Local Speedup: 87.36%

Every scheduler is capable of issuing one instruction per cycle, but for this kernel each scheduler only issues an instruction every 7.9 cycles. This might leave hardware resources underutilized and may lead to less optimal performance. Out of the maximum of 16 warps per scheduler, this kernel allocates an average of 1.36 active warps per scheduler, but only an average of 0.13 warps were eligible per cycle. Eligible warps are the subset of active warps that are ready to issue their next instruction. Every cycle with no eligible warp results in no instruction being issued and the issue slot remains unused. To increase the number of eligible warps, reduce the time the active warps are stalled by inspecting the top stall reasons on the [Warp State Statistics](#) and [Source Counters](#) sections.

Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.

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 x

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 x

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

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 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.

Est. Speedup: 3.05%

L2 Theoretical Sectors Global Excessive

Location	Value	Value (%)
NavStateIndex.h:224 (0xffffcd90b90c0 in void celeritas::detail::<unnamed>::launch...	392,696	2
NavStateIndex.h:236 (0xffffcd90ba400 in void celeritas::detail::<unnamed>::launch...	380,690	2
NavStateIndex.h:143 (0xffffcd90bac80 in void celeritas::detail::<unnamed>::launch...	363,679	2
Transformation3D.h:305 (0xffffcd90ba640 in void celeritas::detail::<unnamed>::lau...	303,195	1
Transformation3D.h:305 (0xffffcd90ba630 in void celeritas::detail::<unnamed>::lau...	303,195	1

Warp Stall Sampling (All Samples)		Most Instructions Executed	
Location	Value	Location	Value
Vector3D.h:380 (0xffffcd90bbef0 in voi...	21,172	BVH.h:117 (0xffffcd90bc3e0 in void cel...	1,409,636
BVH.h:113 (0xffffcd90bbe40 in void cel...	19,782	BVH.h:110 (0xffffcd90bc450 in void cel...	1,373,773
BVH.h:111 (0xffffcd90bbe00 in void cel...	19,322	BVH.h:110 (0xffffcd90bc440 in void cel...	1,373,773
BVH.h:111 (0xffffcd90bbdc0 in void cel...	14,824	BVH.h:110 (0xffffcd90bc430 in void cel...	1,373,773
BVH.h:111 (0xffffcd90bbd80 in void cel...	7,514	BVH.h:110 (0xffffcd90bc420 in void cel...	1,373,773

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.