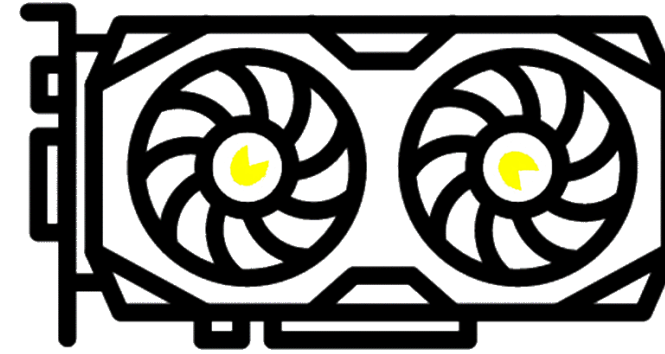
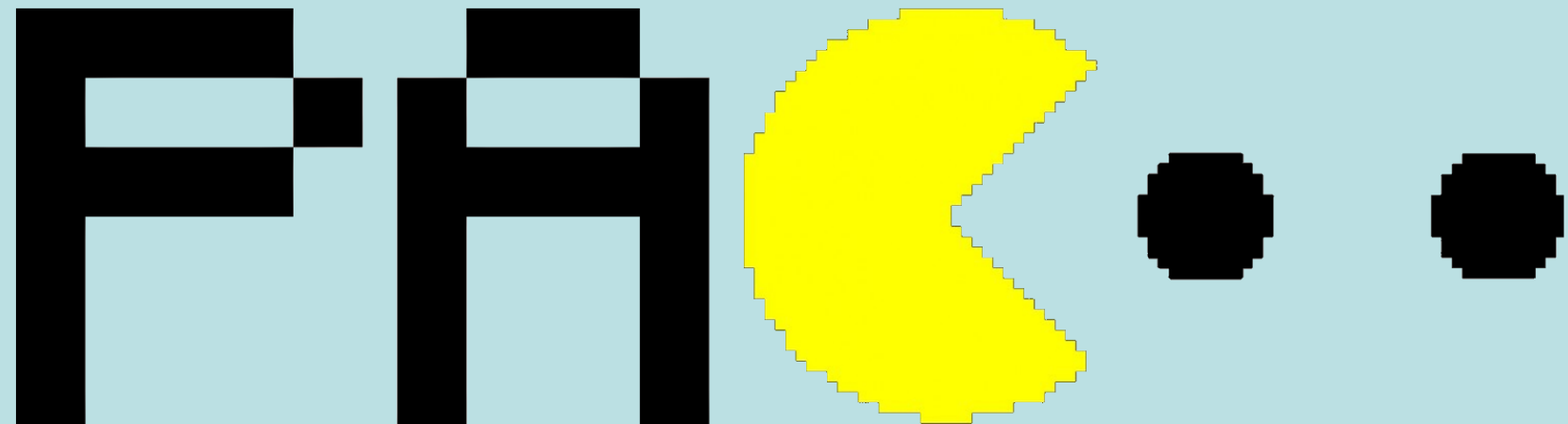


# Parallel Computing



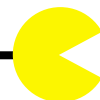
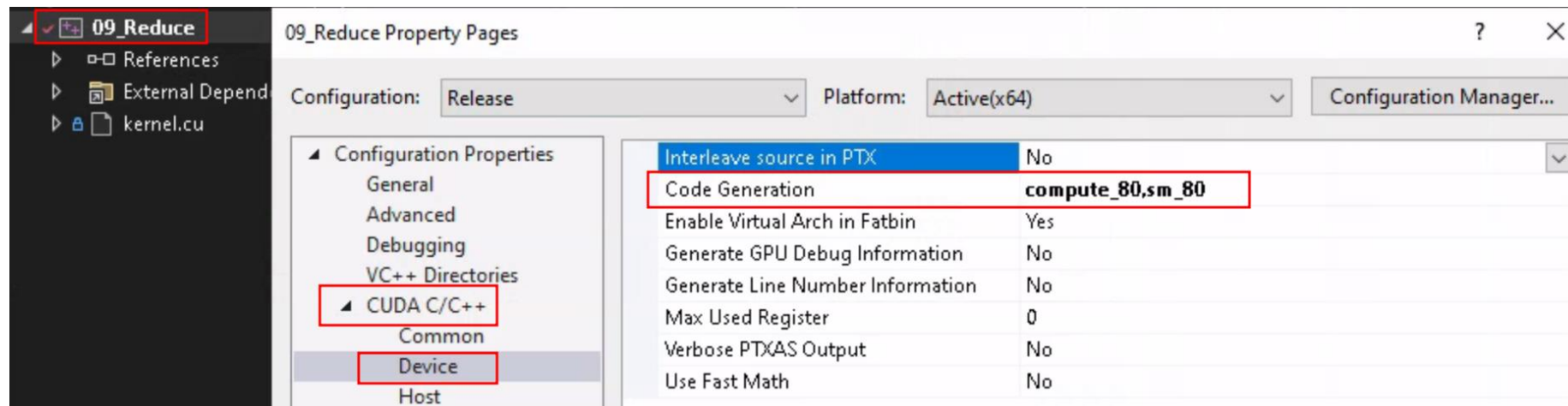
## CUDA Tools



Simon Marcin, Filip Schramka

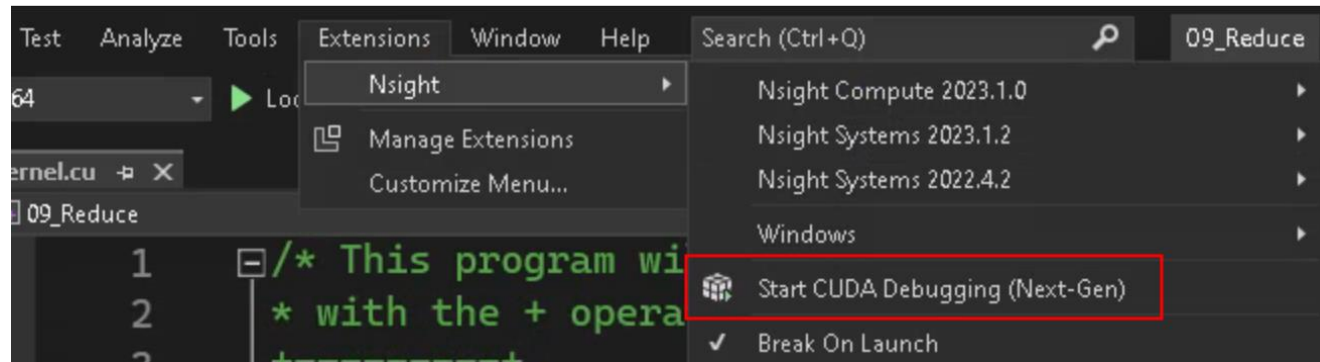
## Visual Studio 2022

- Use for development with a local GPU
  - Nvidia Nsight integration (via VS extension)
  - CUDA Toolkit installation needed
  - Use your own GPU on Windows
- Change the code generation version to match your specific hardware



## Visual Studio 2022

- Start debugging with the CUDA Debugger and not the regular CPU Debugger



- Show CUDA specific debug windows: Extensions -> Nsight -> Windows -> Choose one

## Visual Studio 2022

- Use "Warp Info" to switch between threads/warps/thread blocks.
- Choose a specific thread to show its variables

The screenshot displays two windows from the Visual Studio 2022 IDE. The top window, titled "Autos", shows a list of variables with their values and types. The bottom window, titled "Warp Info", shows a table of thread information with a grid of thread status icons.

**Autos Window:**

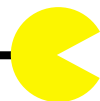
Name	Value	Type
blockDim	{ x = 1024, y = 1, z = 1 }	const dim3
blockDim.x	1024	uint
globalIdx	709	int
shmArray	0x0 { 1 }	__shared__ int [0]
shmArray[threadIdx.x + blockDim.x]	1	int

**Warp Info Window:**

Context	SM Version	Grid ID	Shader Info	Threads
22db70759f0	00080006	00000001	CTA: ( 0, 0, 0), Thread: (608, 0, 0)	[Green icons]
22db70759f0	00080006	00000001	CTA: ( 0, 0, 0), Thread: (640, 0, 0)	[Green icons]
22db70759f0	00080006	00000001	CTA: ( 0, 0, 0), Thread: (672, 0, 0)	[Green icons]
22db70759f0	00080006	00000001	CTA: ( 0, 0, 0), Thread: (704, 0, 0)	[Red icons]

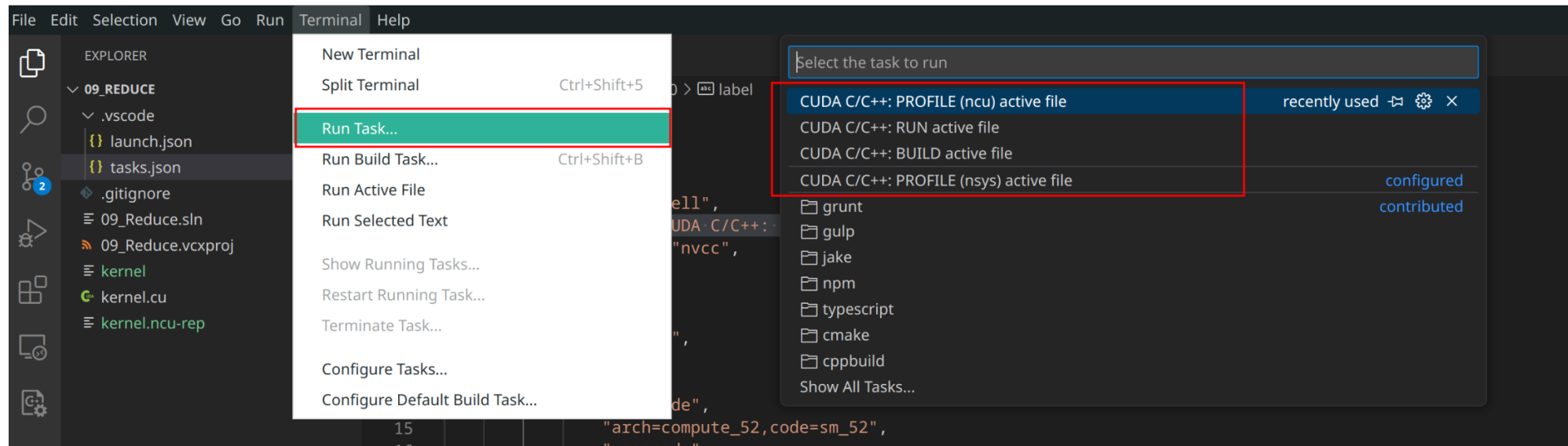
## VS Code

- Use for development with a local or remote GPU
  - Use PAC Ubuntu VM on Paperspace or your own GPU (via WSL)
- Connect to the remote machine (using Remote SSH plugin)
  - `ssh -i pac_key paperspace@publicIP` (this will create a SSH config in VS code for you)
- Install (on the remote session) the Nsight and C++ extension
- Use the tasks.json and launch.json from PAC projects
  - Change tasks.json --> "CUDA C/C++: BUILD active file": Adapt to your hardware
- Intro video: <https://www.youtube.com/watch?v=gN3XeFwZ4ng>



## VS Code

- Use PAC provided Tasks to build, execute or profile your code



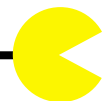
- Profiling tasks will create new files which can be opened with Nsight Tools

## VS Code

- Use "Run --> Start debugging (F5)" to debug CUDA code
- Change the focus (which CUDA thread is shown) by click on the bottom-right corner: CUDA: (0, 0, 0) (0, 0, 0)
- Jump to the CUDA thread of interest (e.g. block(16,0,0) thread(64,0,0))

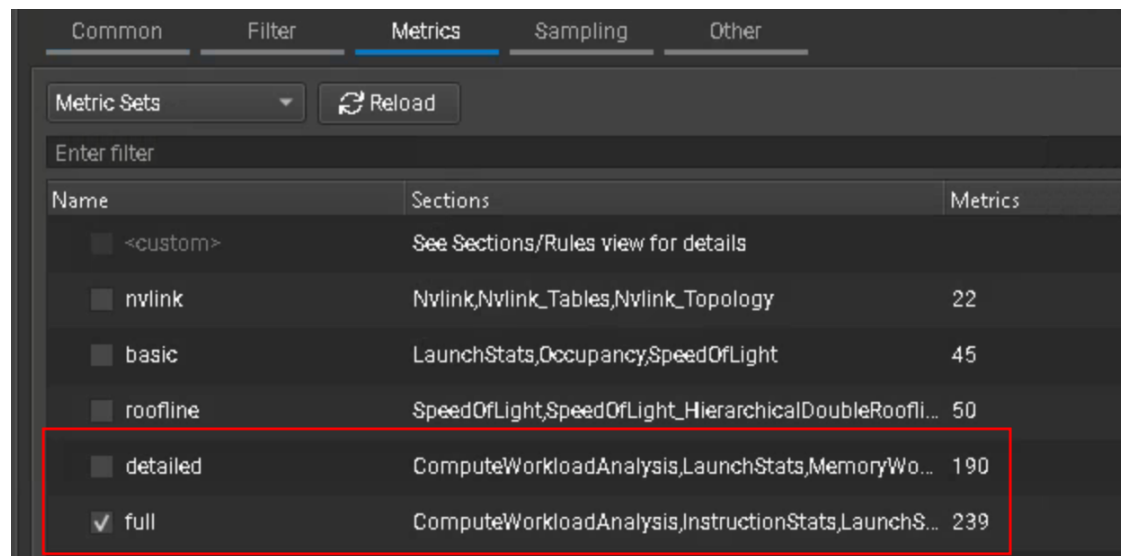


- Use conditional break points to break at the right spot for the right CUDA thread/block



## Nsight Compute

- Use to learn more about the performance of the CUDA kernel
- Choose the metrics you need / are interested

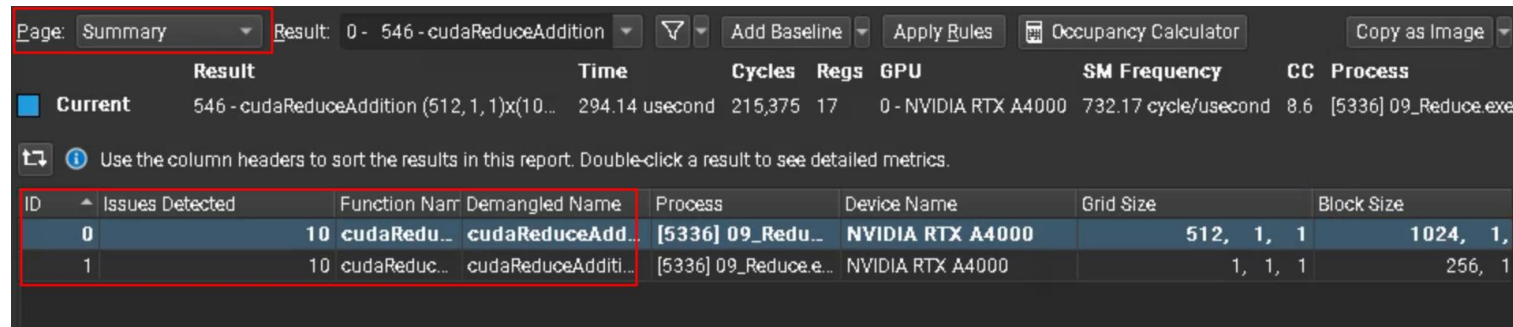


- Nvidia performance counters are need to be enabled for non-root users
- Intro: <https://www.youtube.com/watch?v=04dJ-aePYpE>



## Nsight Compute

- Compare different kernels at once



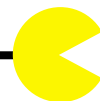
Page: Summary Result: 0 - 546 - cudaReduceAddition Add Baseline Apply Rules Occupancy Calculator Copy as Image

	Result	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	546 - cudaReduceAddition (512,1,1)x(10...	294.14 usecond	215,375	17	0 - NVIDIA RTX A4000	732.17 cycle/usecond	8.6	[5336] 09_Reduce.exe

Use the column headers to sort the results in this report. Double-click a result to see detailed metrics.

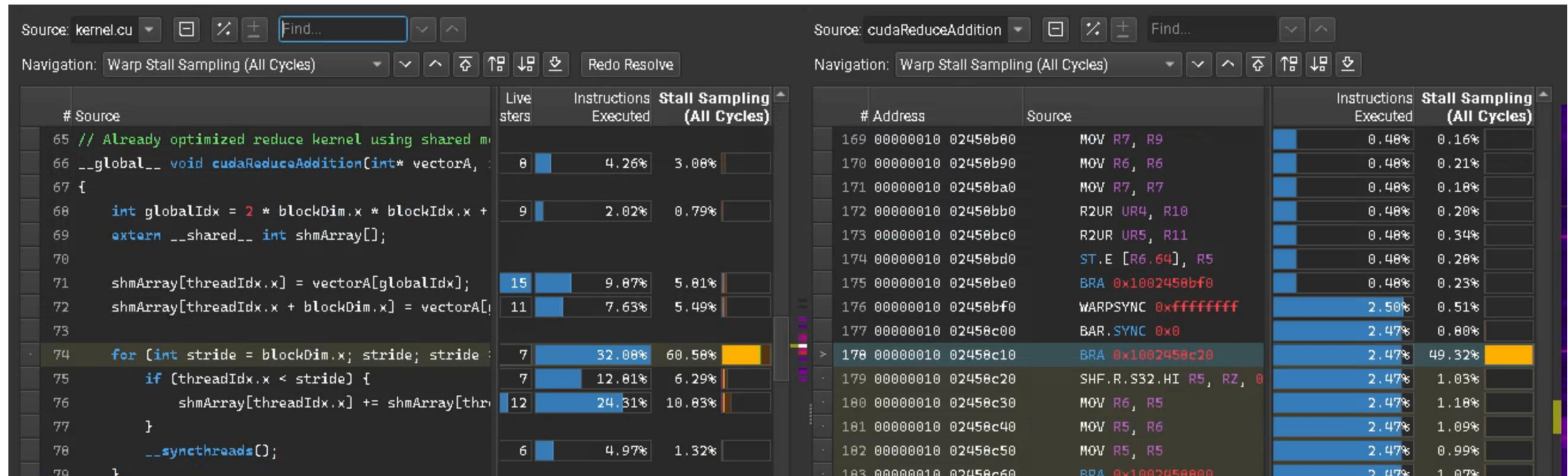
ID	Issues Detected	Function Name	Demangled Name	Process	Device Name	Grid Size	Block Size
0	10	cudaRedu...	cudaReduceAdd...	[5336] 09_Red...	NVIDIA RTX A4000	512, 1, 1	1024, 1,
1	10	cudaReduc...	cudaReduceAdditi...	[5336] 09_Reduce.e...	NVIDIA RTX A4000	1, 1, 1	256, 1

- Check "Occupancy" to see if you hit a sweet spot
- Check "Memory Workload Analysis" to learn more about your data access patterns
  - Coalesced or not
  - Bank conflicts
  - Amount of data transferred
  - Cache misses



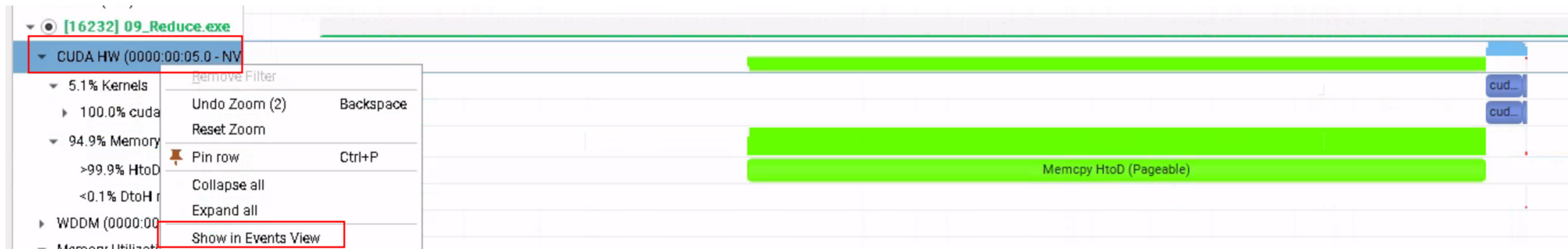
## Nsight Compute

- Use SourceCounters to directly correlate your code with warp stalls and other performance problems
- The warnings often point to this section and provide shortcuts the interesting counters



## Nsight Systems

- Collect information about the timings and duration of the CUDA API calls
- Can you use the memory engine and the kernel engine in parallel?
- Which CUDA stream is execution what and what is overlapping each other



- Select CUDA traces



## Nsight Systems

- The "Event View" will show some basic information about the CUDA API calls and its performance

#	Name	Start	Duration	GPU	Context
1	Memcpy HtoD (Pageable)	0.748356s	2.864 ms	GPU 0	Stream 7
2	cudaReduceAddition(int *, int *)	0.751224s	142.400 µs	GPU 0	Stream 7
3	cudaReduceAddition(int *, int *)	0.751368s	11.200 µs	GPU 0	Stream 7
4	Memcpy DtoH (Pageable)	0.751381s	900 ns	GPU 0	Stream 7

