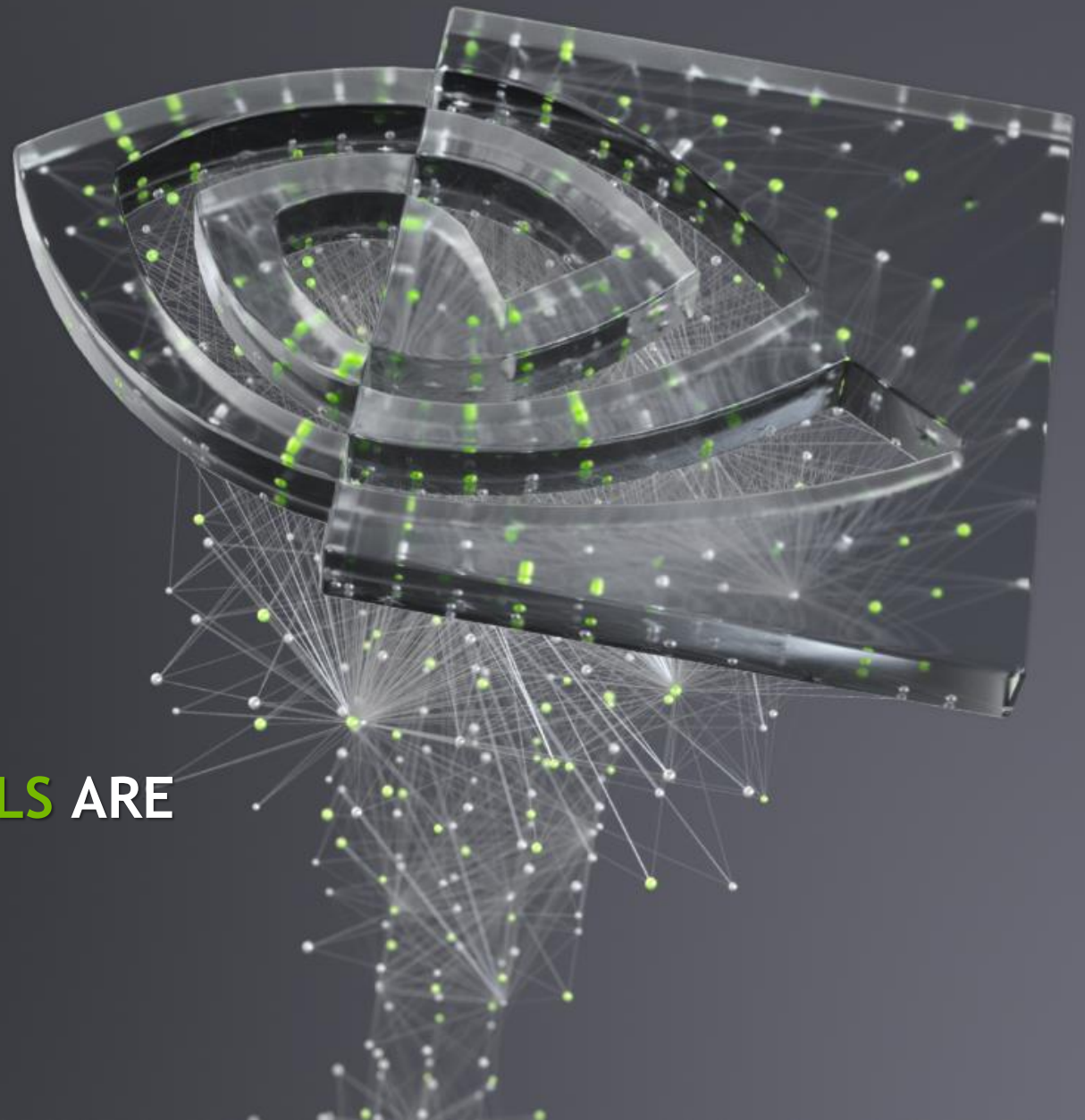
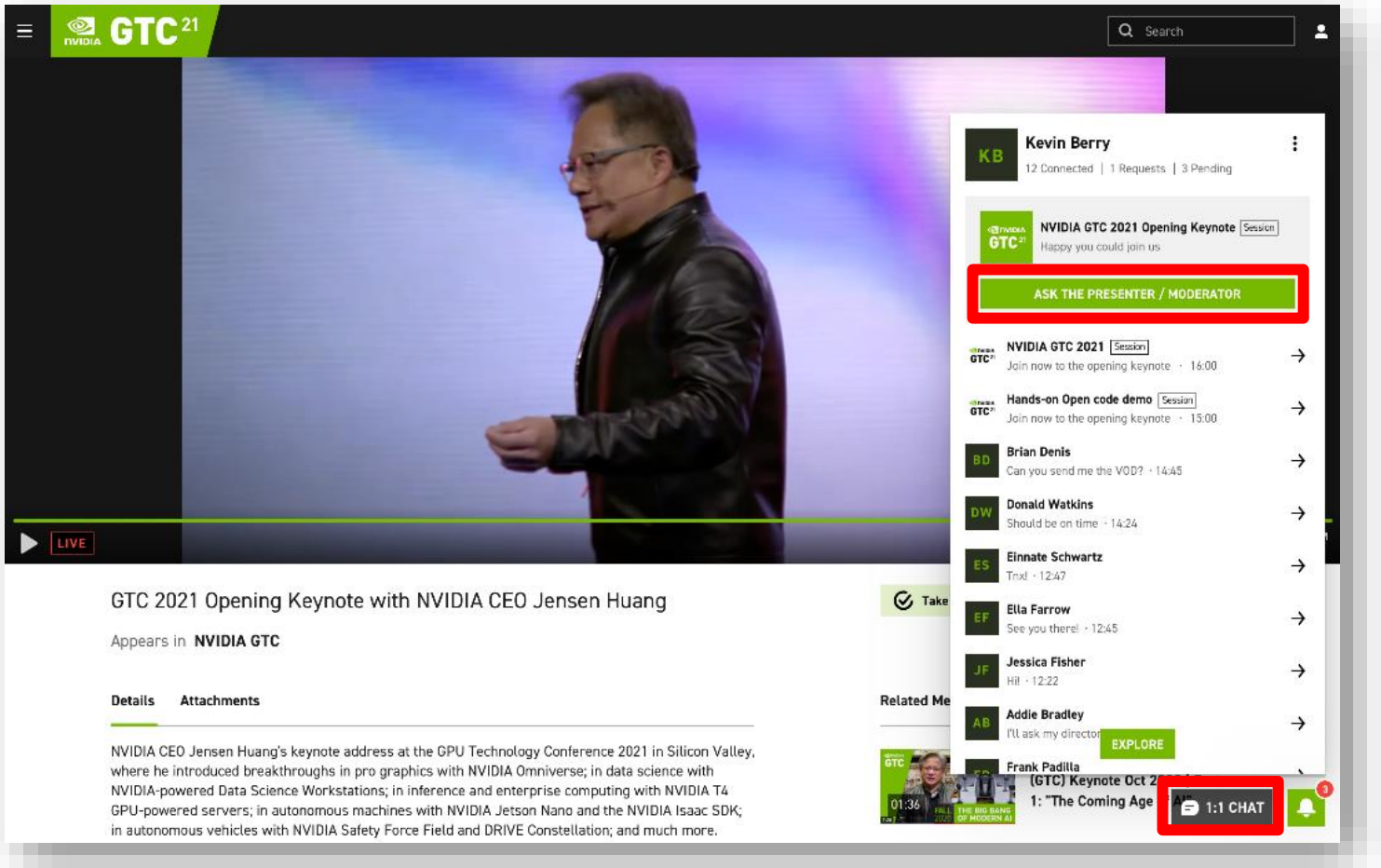
NVIDIA GTC



CUDA IS EVOLVING:
THE LATEST DEVELOPER TOOLS ARE ADAPTING TO KEEP UP

Jackson Marusarz - April 12, 2021

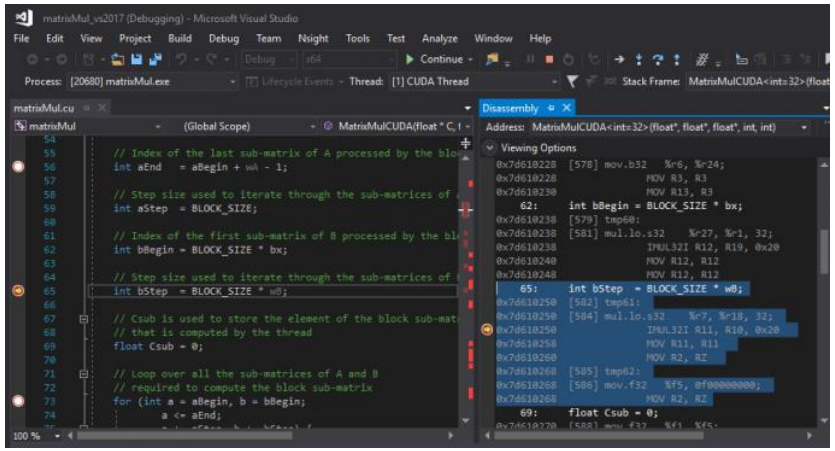
I'M AVAILABLE TO CHAT DURING THIS SESSION



Click on "1:1 Chat," then "Ask the Presenter/Moderator" button to submit your question.
After the session is over, connect with me via attendee chat by searching for my name.

DEVELOPER TOOLS

Debuggers: cuda-gdb, Nsight Visual Studio Edition

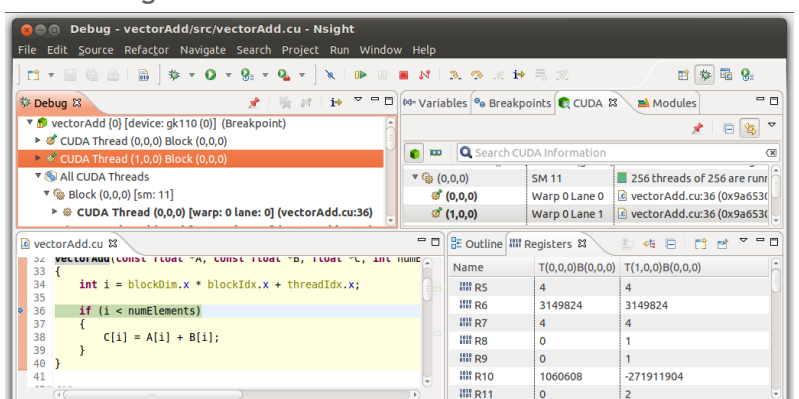


Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)

Correctness Checker: Compute Sanitizer

```
$ compute-sanitizer --leak-check full memcheck_demo
========= COMPUTE-SANITIZER
Mallocing memory
Running unaligned_kernel
Ran unaligned_kernel: no error
Sync: no error
Running out_of_bounds_kernel
Ran out_of_bounds_kernel: no error
Sync: no error
========= Invalid __global__ write of size 4 bytes
=========     at 0x60 in memcheck_demo.cu:6:unaligned_kernel(void)
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x400100001 is misaligned
```

IDE integrations: Nsight Eclipse Edition
Nsight Visual Studio Edition
Nsight Visual Studio **Code** Edition



NVIDIA GTC

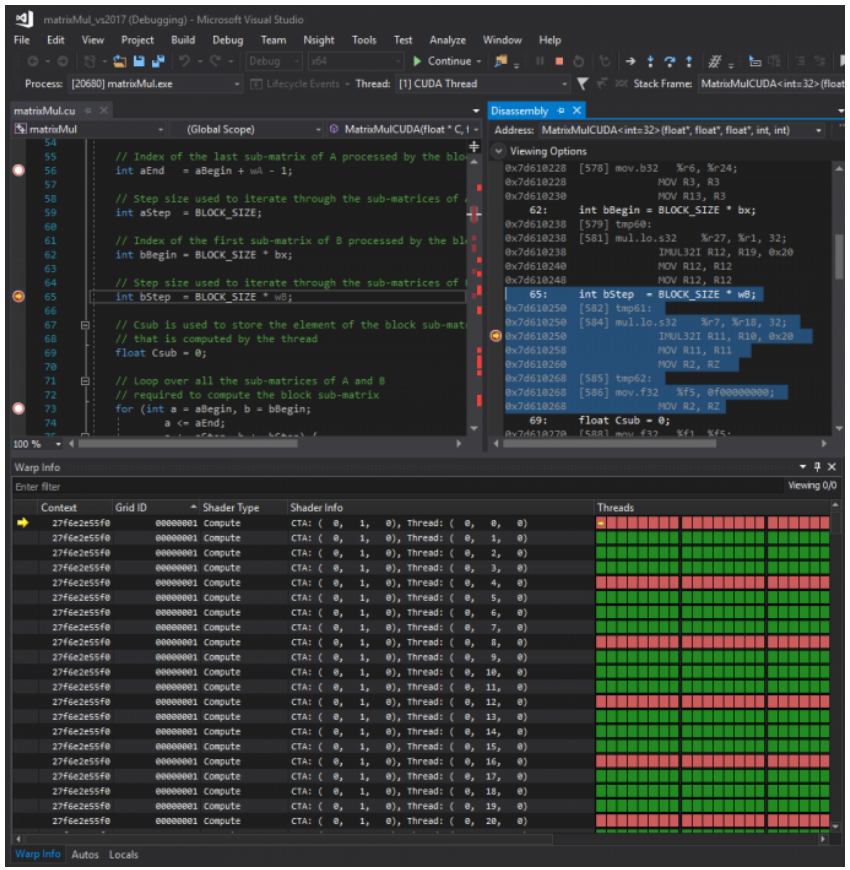

COMPUTE DEBUGGERS
NEW FEATURES

COMPUTE DEBUGGERS

DEBUG GPU KERNELS RUNNING ON DEVICE

Highlights

- Nsight Visual Studio Edition
  - Transitioned to Unified backend (Pascal+)
    - Effort to standardize debuggers
  - Added support for breakpoint "hit count"
  - Added support for display of in-lined functions
  - Implemented "lazy module loading" - provides ~3x performance gain for some workloads
- CUDA GDB
  - Upgraded from GDB 8.2 to 8.3.1
  - Added support for display of in-lined functions



CORRECTNESS CHECKER

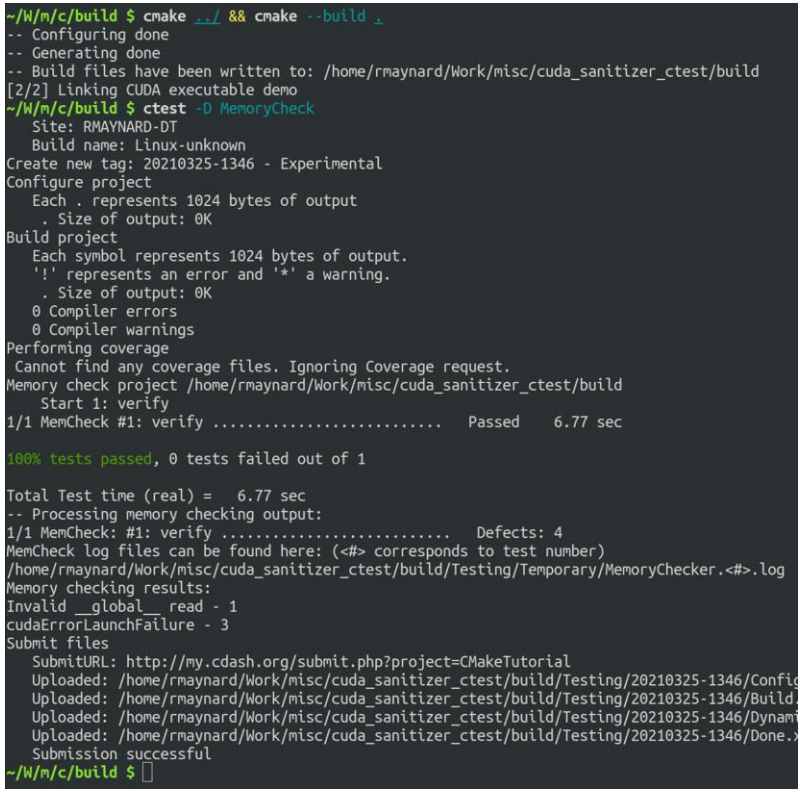
NEW FEATURES

COMPUTE SANITIZER

AUTOMATICALLY SCAN FOR BUGS AND MEMORY ISSUES

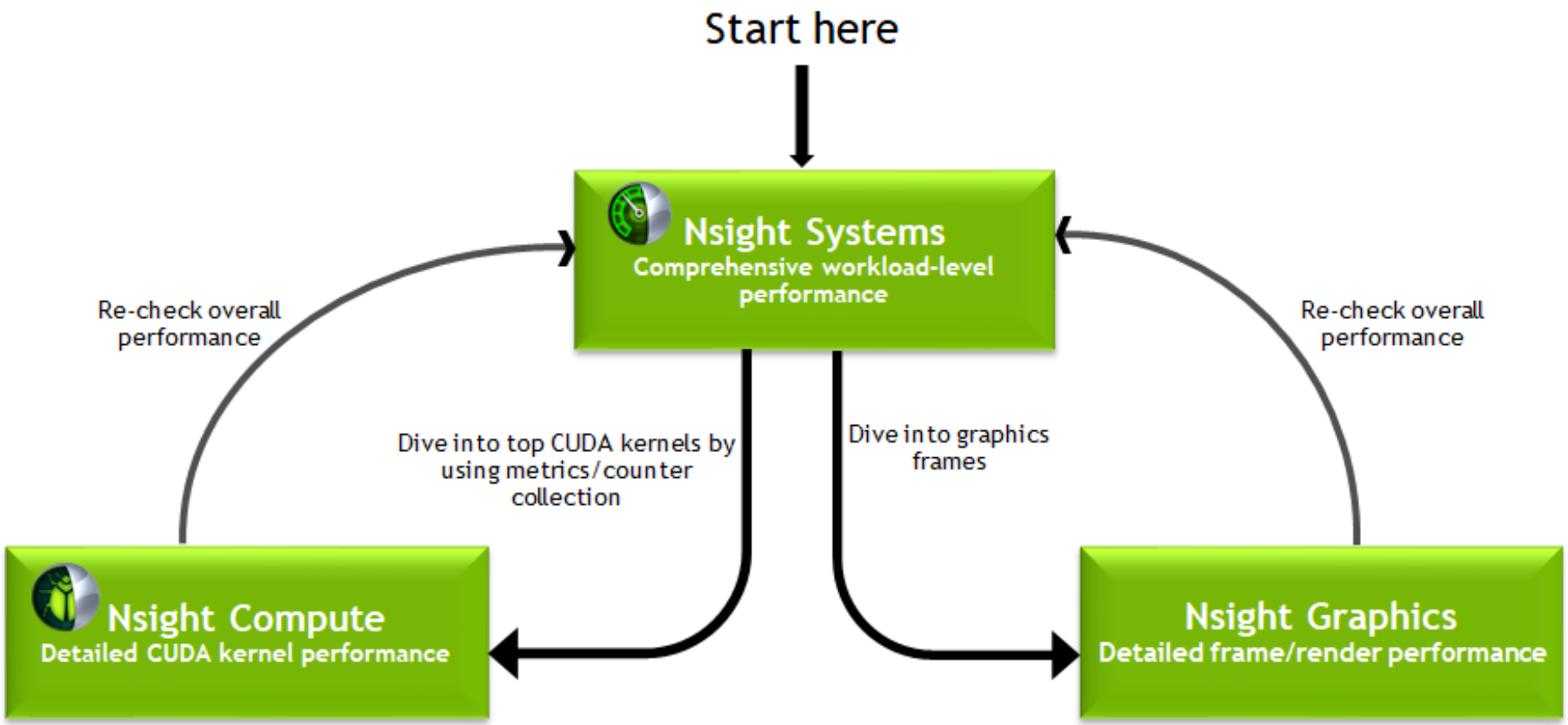
Highlights

- CUDA graphs support
- Experimental support for NVTX memory API to create custom memory pools
  - Allows use of sanitizer tools when using a custom allocator
  - Support for allocation permissions and allocation naming
- Support for allocation padding
  - Improves off-by-one error detection for contiguous allocations
- CMake 3.19 supports Compute Sanitizer analysis
  - Automatically scan for errors '`ctest -D MemoryCheck`'



NSIGHT PROFILING TOOLS WORKFLOW

ITERATIVE OPTIMIZATION FOR COMPUTE AND GRAPHICS




NSIGHT SYSTEMS

NSIGHT SYSTEMS

SYSTEM PROFILER

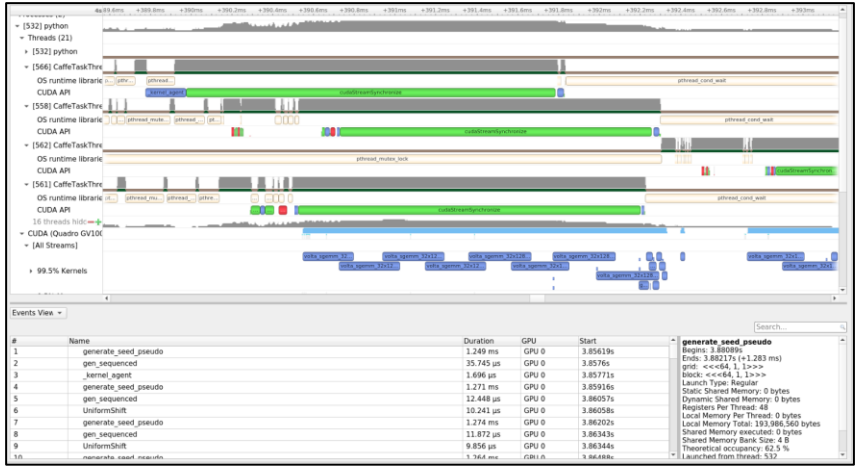
Key Features:

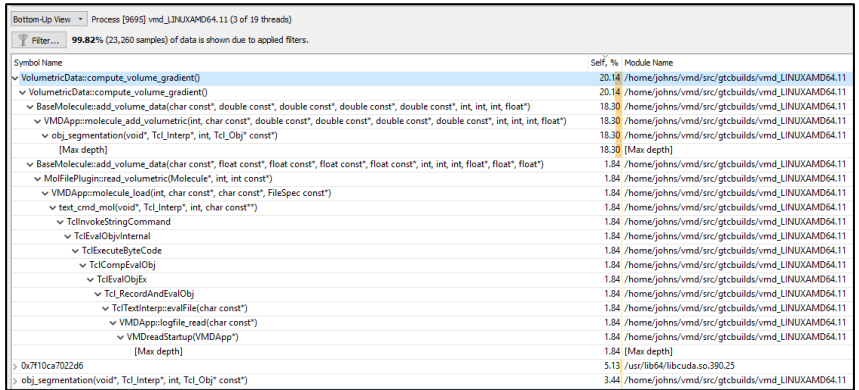
- System-wide application algorithm tuning
  - Multi-process tree support
- Locate optimization opportunities
  - Visualize millions of events on a very fast GUI timeline
  - Or gaps of unused CPU and GPU time
- Balance your workload across multiple CPUs and GPUs
  - CPU algorithms, utilization and thread state
    GPU streams, kernels, memory transfers, etc
- Command Line, Standalone, IDE Integration

OS: Linux (x86, Power, Arm SBSA, Tegra), Windows, MacOSX (host)

GPUs: Pascal+

Docs/product: https://developer.nvidia.com/nsight-systems

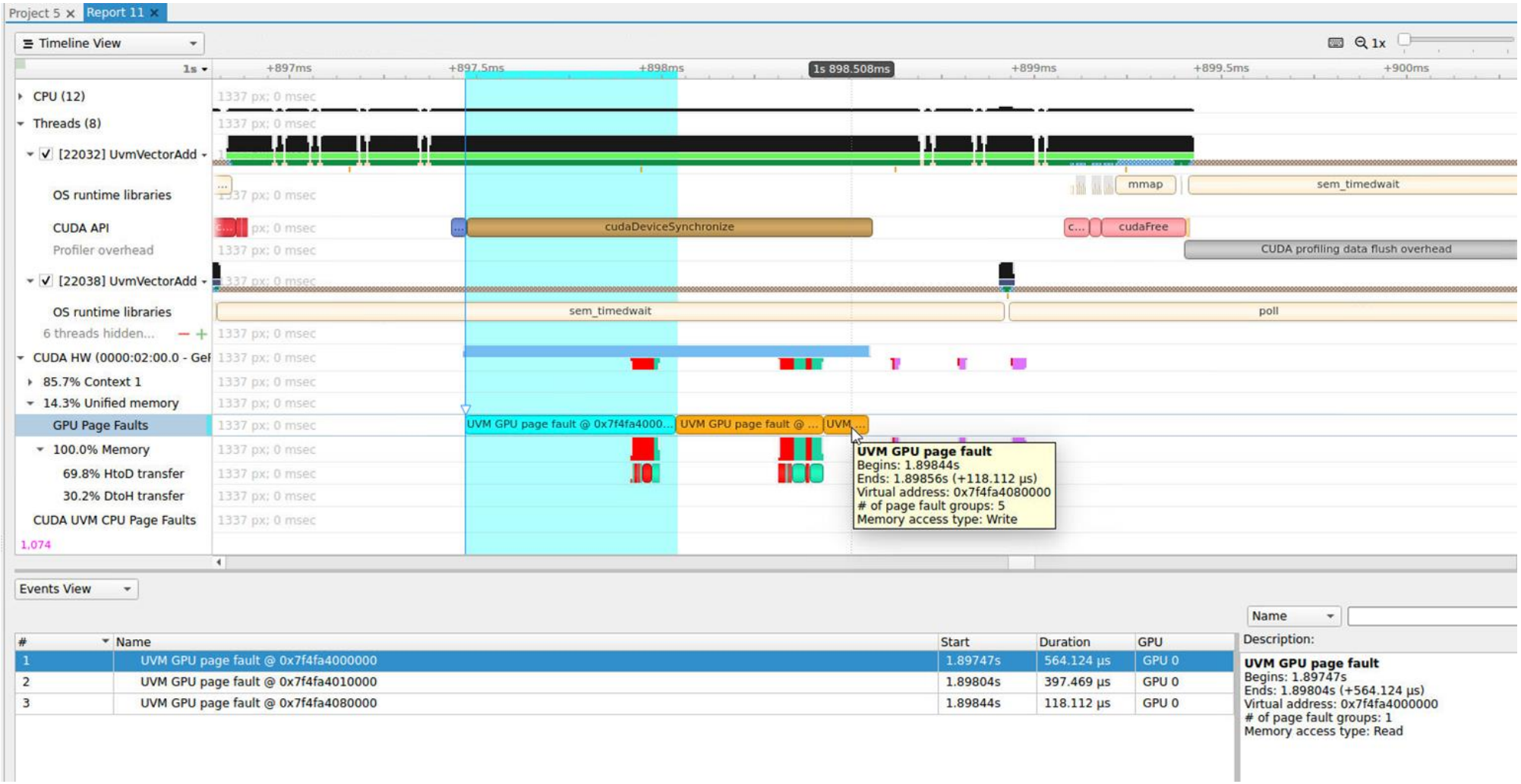





NSIGHT SYSTEMS
NEW FEATURES

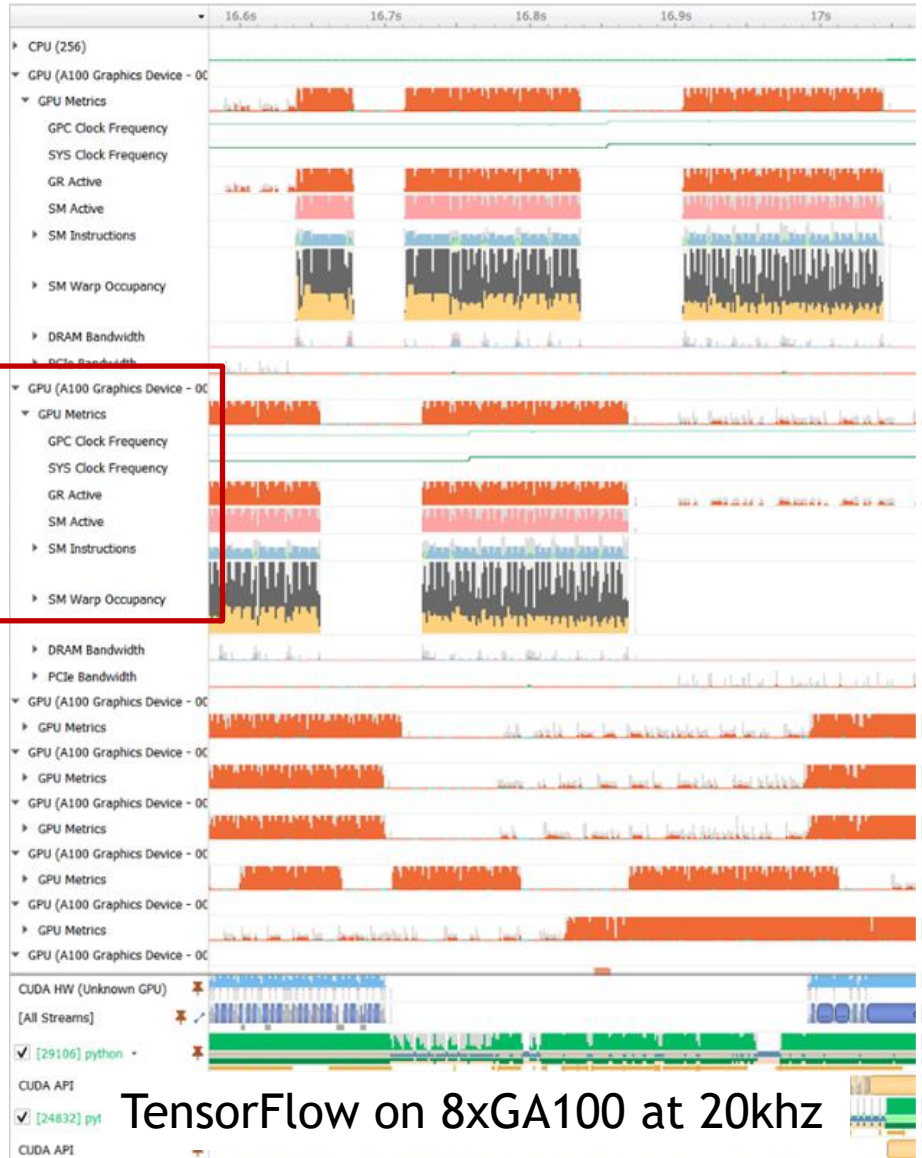
CUDA UNIFIED MEMORY CPU & GPU PAGE FAULT TRACE

FIND PERFORMANCE IMPACTS FROM PAGING



#	Name	Start	Duration	GPU
1	UVM GPU page fault @ 0x7f4fa4000000	1.89747s	564.124 µs	GPU 0
2	UVM GPU page fault @ 0x7f4fa4010000	1.89804s	397.469 µs	GPU 0
3	UVM GPU page fault @ 0x7f4fa4080000	1.89844s	118.112 µs	GPU 0

GPU METRICS SAMPLING



TensorFlow on 8xGA100 at 20khz

- Is my GPU full? Sufficient grid size & streams?
- Is my instruction rate low (possibly I/O bound)?
- Am I using TensorCores?
- Can I see GPU Direct RDMA|Storage or other transfers?
- System-wide GPU observation (no app required, but sudo or regkey)
- 10khz default can be increased depending on GPU
- Metrics:

SM utilizations:

- **SMs active**
- **Instructions**
- **TensorCores**
- **Warp occupancy**
  - **Unallocated slots**

IO throughputs:

- **PCIe**
- **NVLink**
- **DRAM**


NSIGHT COMPUTE

NSIGHT COMPUTE

KERNEL PROFILING TOOL

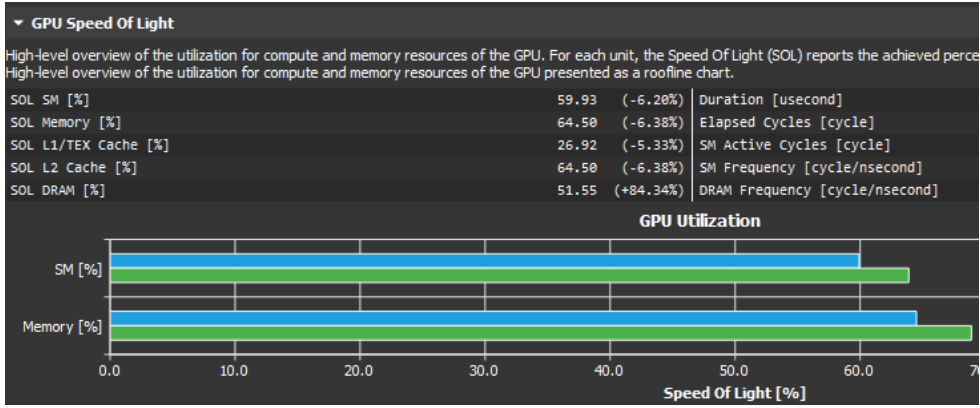
Key Features:

- Interactive CUDA API debugging and kernel profiling
- Built-in rules expertise
- Fully customizable data collection and display
- Command Line, Standalone, IDE Integration, Remote Targets

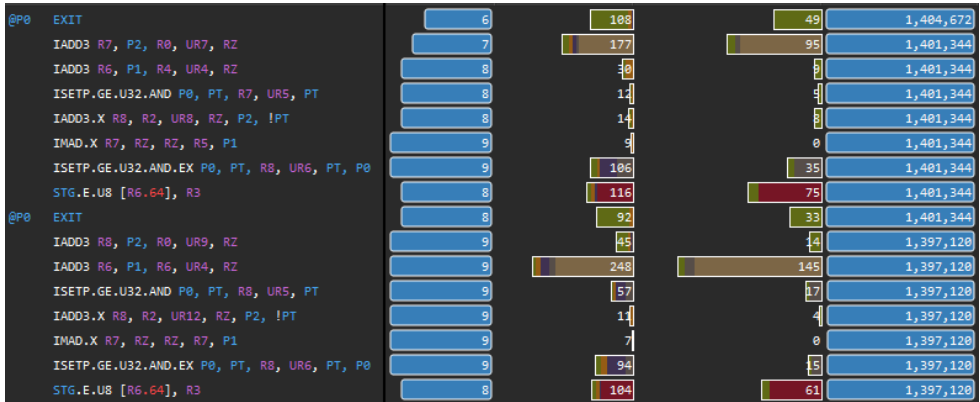
OS: Linux (x86, Power, Tegra, Arm SBSA), Windows, MacOSX (host only)

GPUs: Volta, Turing, Ampere GPUs

Docs/product: https://developer.nvidia.com/nsight-compute



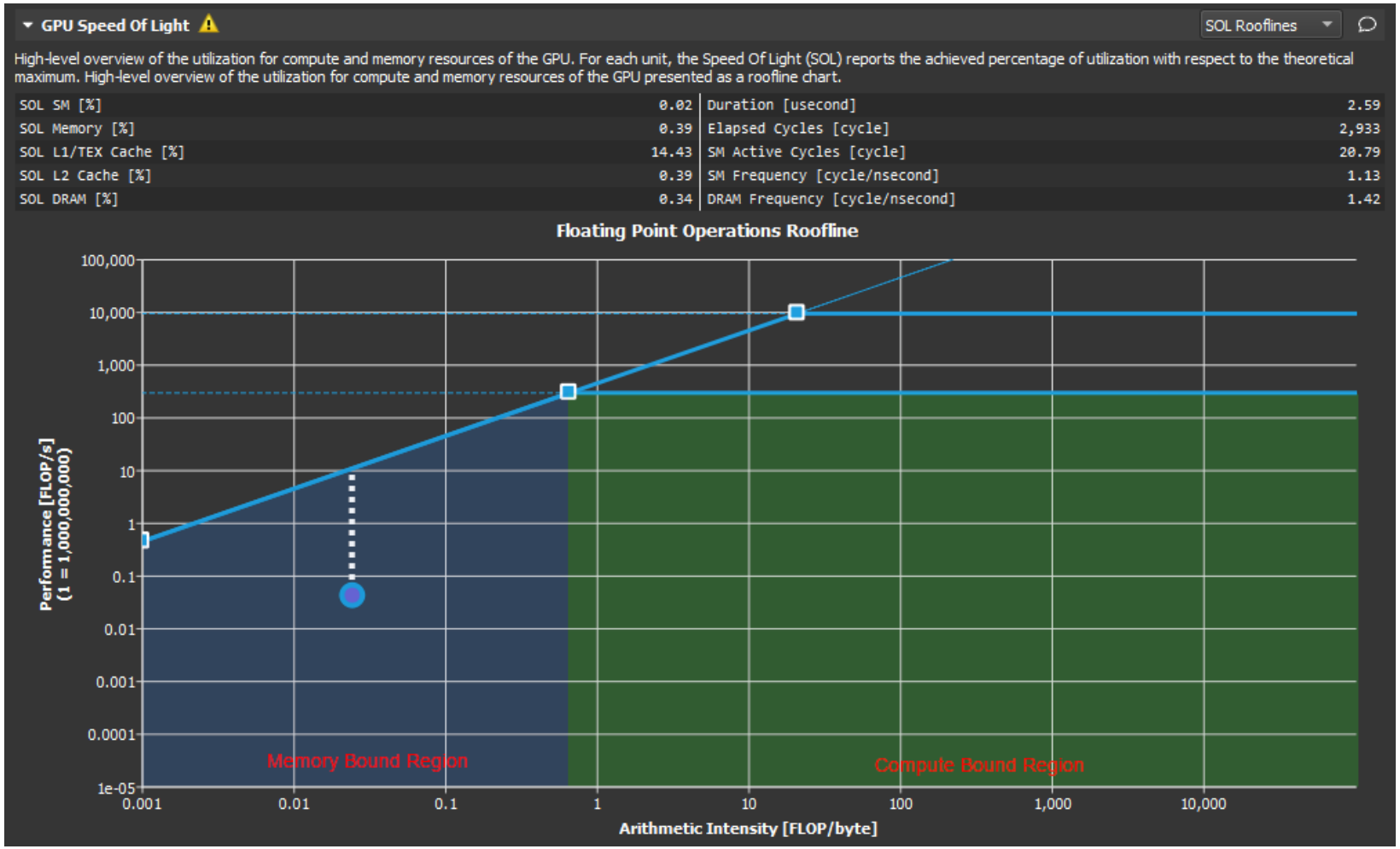
inst_executed [inst]	63,021,056 (284 instances)
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.sum	0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.sum	0
l1tex__data_bank_reads.avg.pct_of_peak_sustained_elapsed [%]	9.66
l1tex__data_bank_writes.avg.pct_of_peak_sustained_elapsed [%]	3.23
l1tex__data_pipe_lsu_wavefronts.avg.pct_of_peak_sustained_elapsed [%]	46.16
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_read.sum	25,165,824
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_read.sum.pct_of_peak_sustained_active [%]	40.75
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_write.sum	2,097,152
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_write.sum.pct_of_peak_sustained_active [%]	3.40
l1tex__data_pipe_tex_wavefronts.avg.pct_of_peak_sustained_elapsed [%]	0
l1tex__f_wavefronts.avg.pct_of_peak_sustained_elapsed [%]	0.00
l1tex__lsu_writeback_active.avg.pct_of_peak_sustained_elapsed [%]	42.59
l1tex__lsu_writeback_active.sum [cycle]	27,803,648
l1tex__lsu_writeback_active.sum.pct_of_peak_sustained_active [%]	45.03
l1tex__lsuin_requests.avg.pct_of_peak_sustained_elapsed [%]	66.00
l1tex__m_l1tex2xbar_req_cycles_active.avg.pct_of_peak_sustained_elapsed [%]	3.40
l1tex__m_l1tex2xbar_write_bytes.sum [Mbyte]	4.19
l1tex__m_l1tex2xbar_write_bytes_mem_global_op_red.sum [byte]	0




NSIGHT COMPUTE
NEW FEATURES

ROOFLINE ANALYSIS

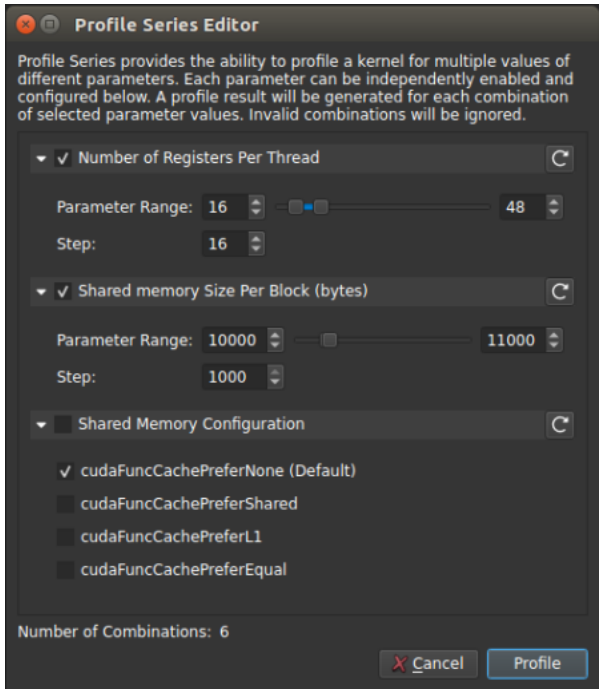
FIND HARDWARE LIMITATIONS FOR CUDA KERNELS



- Graph Performance vs. Arithmetic Intensity
- Identify memory or compute bottlenecks for each kernel
- Built-in "roofs" represent system hardware limitations

PROFILE SERIES (WHAT-IF)

AUTOMATIC PERFORMANCE ANALYSIS OF CONFIGURABLE PARAMETERS



Profile series repeatedly profiles a kernel with a range of configurable parameters to analyze the performance of each combination.

- Number of register per thread
- Shared memory size per block
- Shared memory configuration

Page: Summary Launch: 4 - 18 - VecAdd_kernel Add Baseline Apply Rules

Current 18 - VecAdd_kernel (196, 1, 1)x(256, 1, 1) **Time:** 4.86 usecond **Cycles:** 6,007 **Regs:** 16 **GPU:** GeForce RTX 2080 Ti **SM Frequency:** 1.22 cycle/nsecond **CC:** 7.5 **Process:** [27553] CuVectorAddDrv

ID	Series ID	Series Parameters	Time	API Call ID	Function Name	Der	Process	Device Name	Cycles [cycle]	Duration [usecond]
0	0	default	2021-Mar-01 11:31:…	18	VecAdd_kernel		[27553] CuVect…	GeForce RTX 2080 Ti	6,104	5.22
1	0	num_reg_per_thread:16	2021-Mar-01 11:31:…	18	VecAdd_kernel		[27553] CuVect…	GeForce RTX 2080 Ti	6,138	5.01
2	0	num_reg_per_thread:32	2021-Mar-01 11:31:…	18	VecAdd_kernel		[27553] CuVect…	GeForce RTX 2080 Ti	6,048.33	5.07
3	0	num_reg_per_thread:48	2021-Mar-01 11:31:…	18	VecAdd_kernel		[27553] CuVect…	GeForce RTX 2080 Ti	6,046	4.99
4	1	default	2021-Mar-01 11:32:…	18	VecAdd_kernel		[27553] CuVect…	GeForce RTX 2080 Ti	6,007	4.86
5	1	num_reg_per_thread:16,dynamic_shared_mem_per_block:10000	2021-Mar-01 11:32:…	18	VecAdd_kernel		[27553] CuVect…	GeForce RTX 2080 Ti	5,925	5.25
6	1	num_reg_per_thread:16,dynamic_shared_mem_per_block:11000	2021-Mar-01 11:32:…	18	VecAdd_kernel		[27553] CuVect…	GeForce RTX 2080 Ti	5,952.67	5.09
7	1	num_reg_per_thread:32,dynamic_shared_mem_per_block:10000	2021-Mar-01 11:32:…	18	VecAdd_kernel		[27553] CuVect…	GeForce RTX 2080 Ti	5,921	5.11

NEW NVLINK SECTION

UNDERSTAND THE INTERCONNECT PERFORMANCE

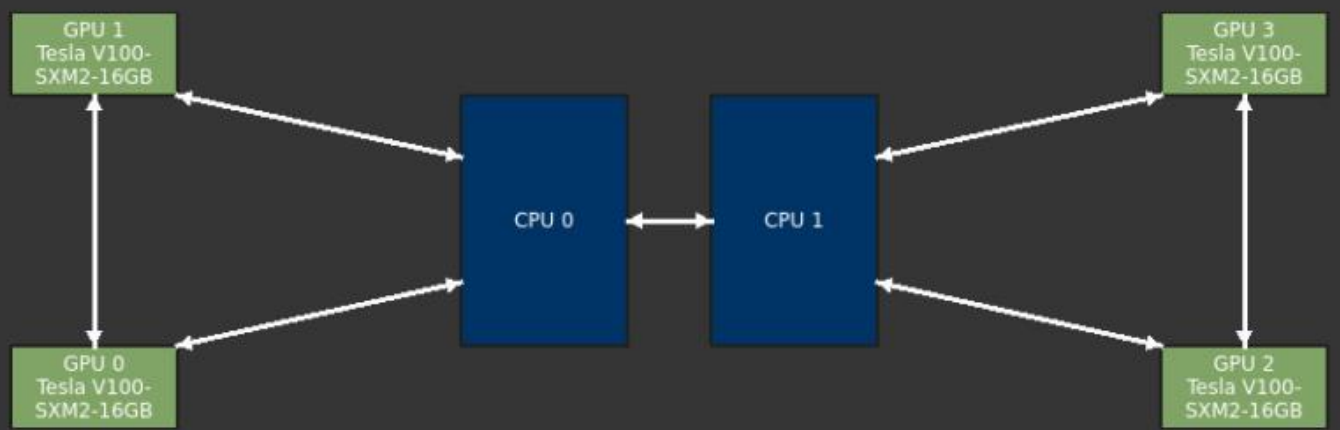
Device-level Metrics

NVLink All

High-level summary of NVLink utilization. It shows the total received and transmitted (sent) memory, as well as the overall link peak utilization. NVLink Topology. Detailed tables with properties for each NVLink.

Physical Links	30	Logical Links	10
Received Bytes [Kbytes]	927.06	Received Peak Utilization [%]	0.08
Transmitted Bytes [Mbytes]	1.09	Transmitted Peak Utilization [%]	0.10
Received Overhead Bytes [Kbytes]	438.69	Transmitted Overhead Bytes [Mbytes]	1.09
Received User Bytes [Kbytes]	488.38	Transmitted User Bytes [Kbytes]	4.72

NVLink Topology



Platform Topology

Logical NVLink Properties

Logical NVLink	Peak Bandwidth	Physical NVLinks	Peer Access	System Access	Peer Atomic	System Atomic
GPU 0<-->GPU 1	157.286 GB/s	3	Yes	No	Yes	No
GPU 0<-->GPU 2	88.0804 GB/s	3	Yes	No	Yes	No
GPU 0<-->GPU 3	88.0804 GB/s	3	Yes	No	Yes	No
GPU 1<-->GPU 2	88.0804 GB/s	3	Yes	No	Yes	No

Per-Link Information


CUPTI
CUDA PROFILING TOOLS INTERFACE

CUPTI

Highlights

New low-overhead GPU PC Sampling APIs (Production CUDA 11.3 GA)

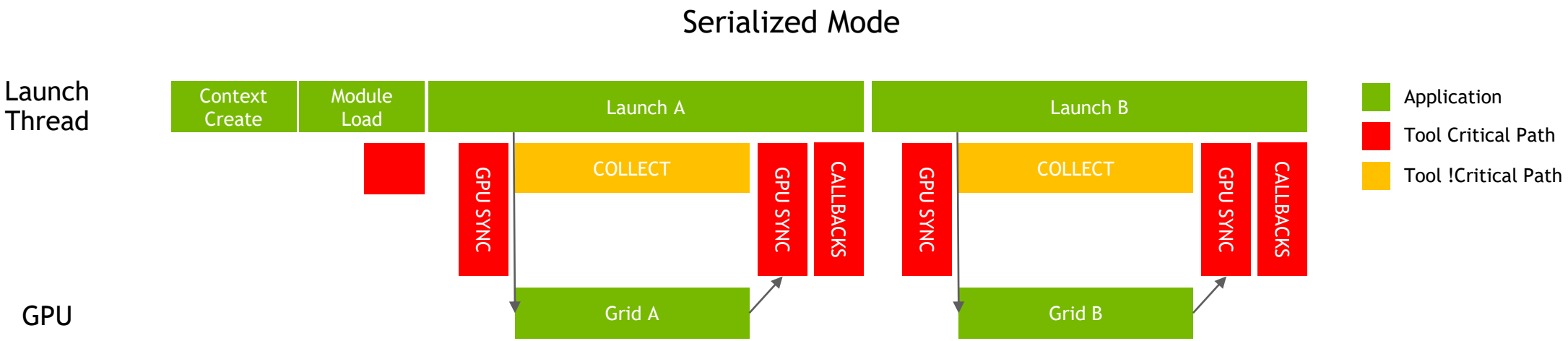
- Collects GPU Program Counter (PC) and Stall reason (if applicable)
- Sampling modes – Continuous (concurrent kernels) and Serialized (sequential kernels)
    - Performance Numbers for continuous mode on Volta GPU:
        - Single GPU: <2% for sampling period 11 (2^11 cycles) on CuLaunches, raja-perf, GROMACS
        - 2 GPU: <2% 4 GPU <4% - sampling period 15 (GROMACS)
- APIs support offline and runtime correlation of PC samples to CUDA source/GPU assembly
- Volta+ GPUs supported on Linux (x86, ppc64le) and Windows
- HPC Toolkit and LBNL collaboration

Correlation of Graph nodes with Graph and GPU activities

Tracing performance improvements

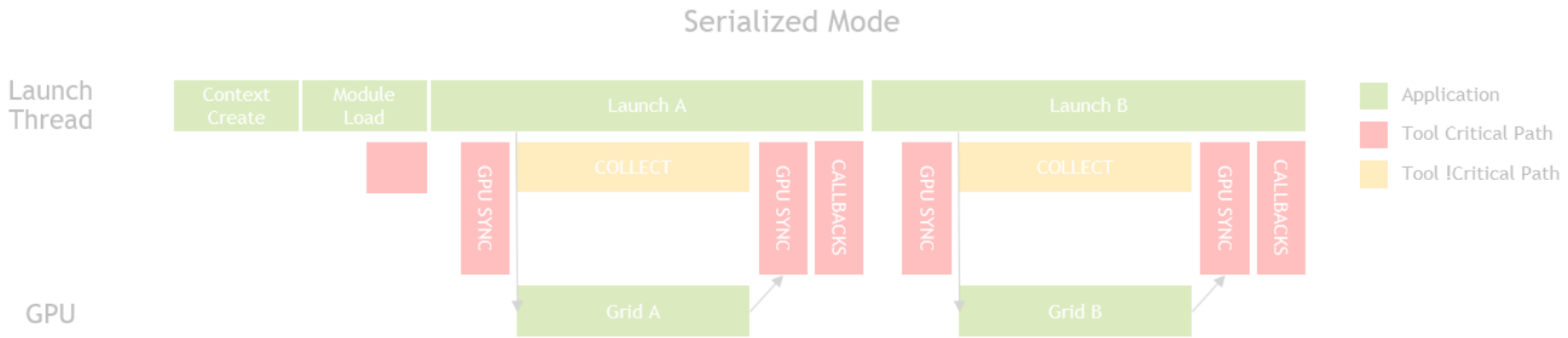
PC SAMPLING MODES

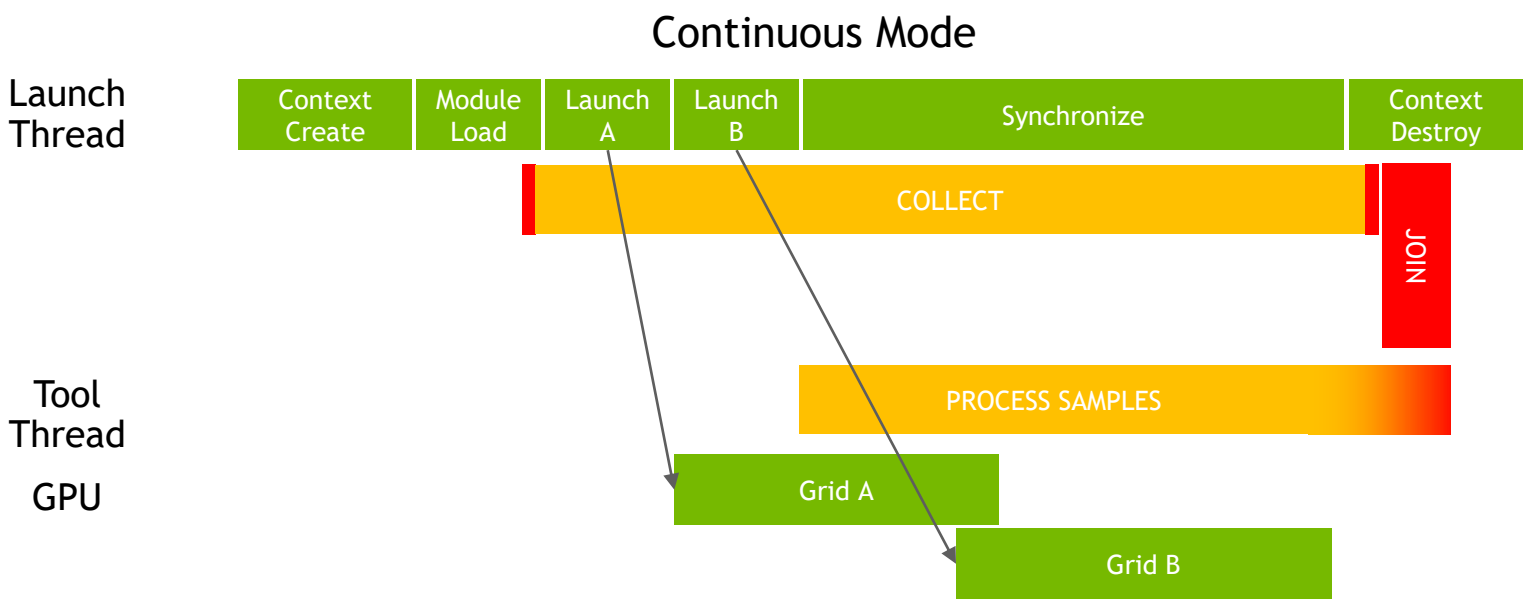
SERIALIZED KERNELS LIMIT PERFORMANCE



PC SAMPLING MODES

CONTINUOUS SAMPLING FOR BETTER REALISTIC PERFORMANCE






NSIGHT VISUAL STUDIO CODE EDITION

NSIGHT VISUAL STUDIO CODE EDITION

A NEW WAY TO DEVELOP AND DEBUG CUDA

Partnership with Microsoft to enable CUDA as a first-class citizen

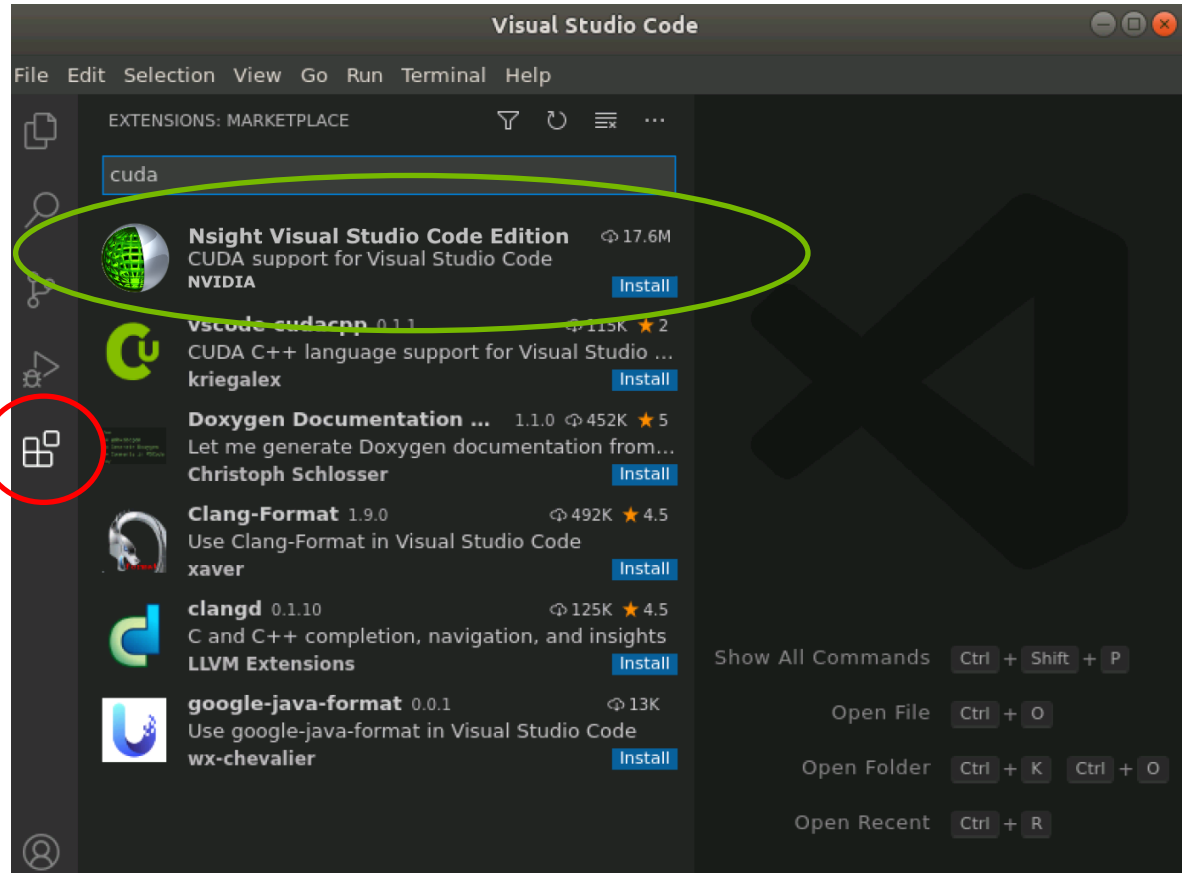
Visual Studio Code extensions that provide:

- CUDA code syntax highlighting
- CUDA code completion
- Build warning/errors
- Debug CPU & GPU code

https://developer.nvidia.com/nsight-visual-studio-code-edition

HOW WILL I GET NSIGHT VS CODE EDITION?

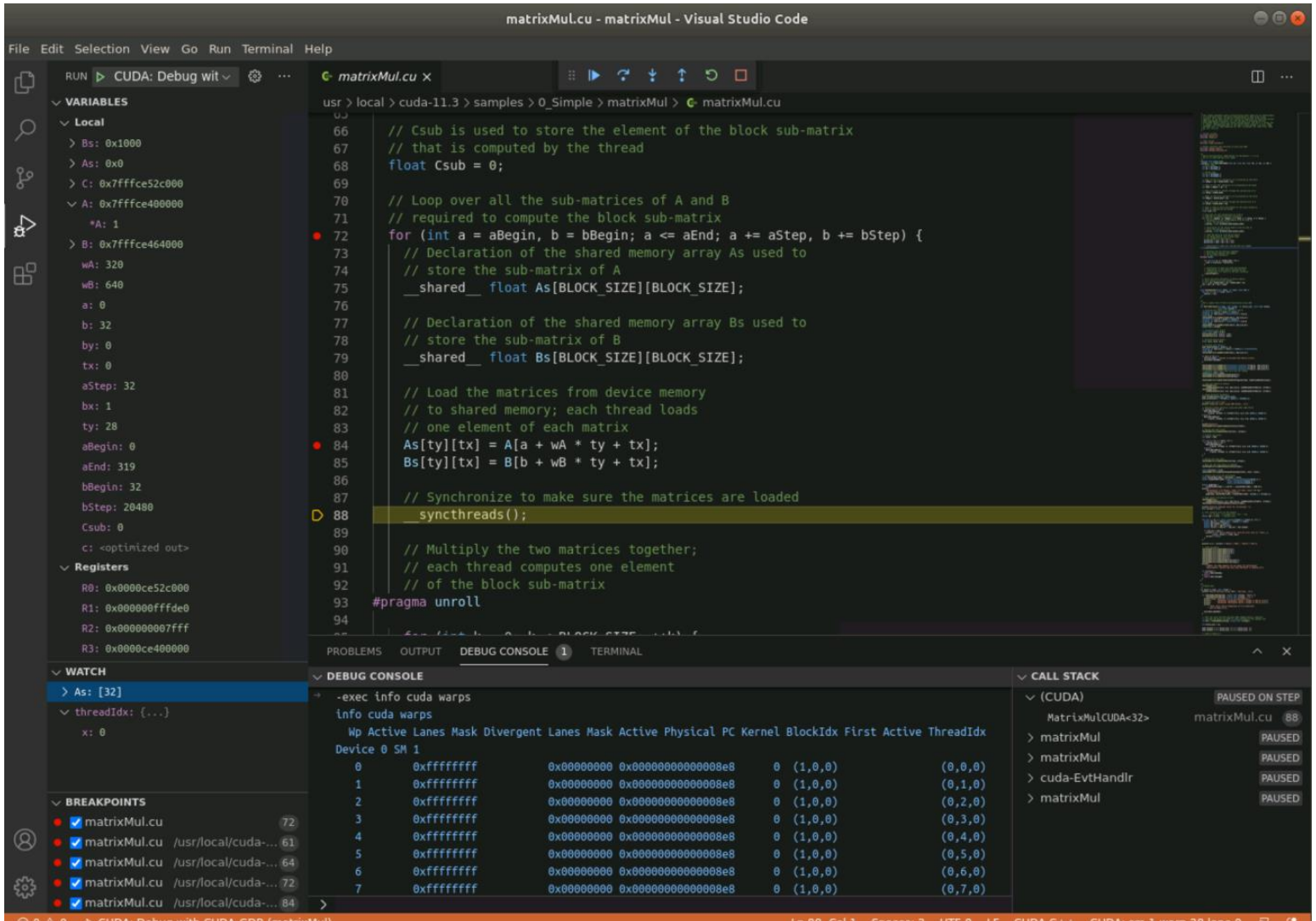
Ease of discoverability & installation

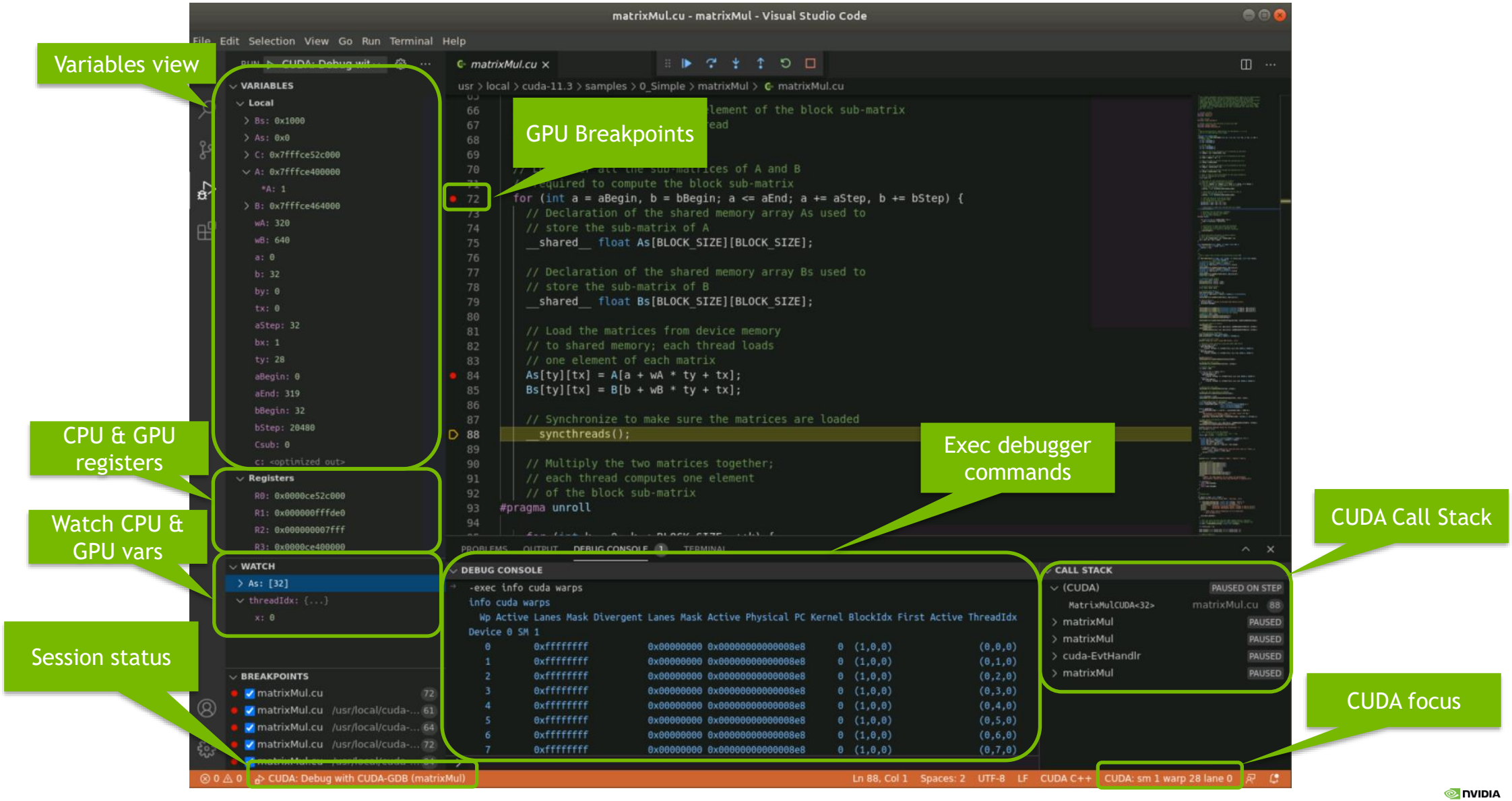


Options for installation:

1. Built-in Visual Studio Code “Marketplace”
2. Download VSIX from VS Code “Marketplace” site – manual install

VS CODE DEBUGGING




matrixMul.cu - matrixMul - Visual Studio Code
Variables view
GPU Breakpoints
CPU & GPU registers
Watch CPU & GPU vars
Exec debugger commands
CUDA Call Stack
Session status
CUDA focus
VARIABLES
Local
Registers
WATCH
BREAKPOINTS
DEBUG CONSOLE
CALL STACK
CUDA: Debug with CUDA-GDB (matrixMul)
CUDA: sm 1 warp 28 lane 0

CALL TO ACTION

- Developer Tools are free and packaged in the 11.3 version of the CUDA Toolkit
  - https://developer.nvidia.com/cuda-downloads
- Most recent versions can also be found in the download center
  - Nsight Systems GPU Metrics are in the latest version here
  - https://developer.nvidia.com/gameworksdownload
- Support is available via:
  - https://forums.developer.nvidia.com/c/development-tools/
- More information at:
  - https://developer.nvidia.com/tools-overview

MORE TO DO @ GTC

GTC Sessions

Performance Tuning CUDA Applications with the Roofline Model [S32062]

Requests, Wavefronts, Sectors Metrics: Understanding and Optimizing Memory-Bound Kernels with Nsight Compute [S32089]

CUDA is Evolving, and the Latest Developer Tools are Adapting to Keep Up [S31747]

Tuning GPU Network and Memory Usage in Apache Spark [31566]

Latest Enhancements to CUDA Debugger IDEs [S31884]

GTC Labs

Optimizing CUDA Machine Learning Codes with Nsight Profiling Tools [T2503]

Debugging and Analyzing Correctness of CUDA Applications [T2504]


NVIDIA