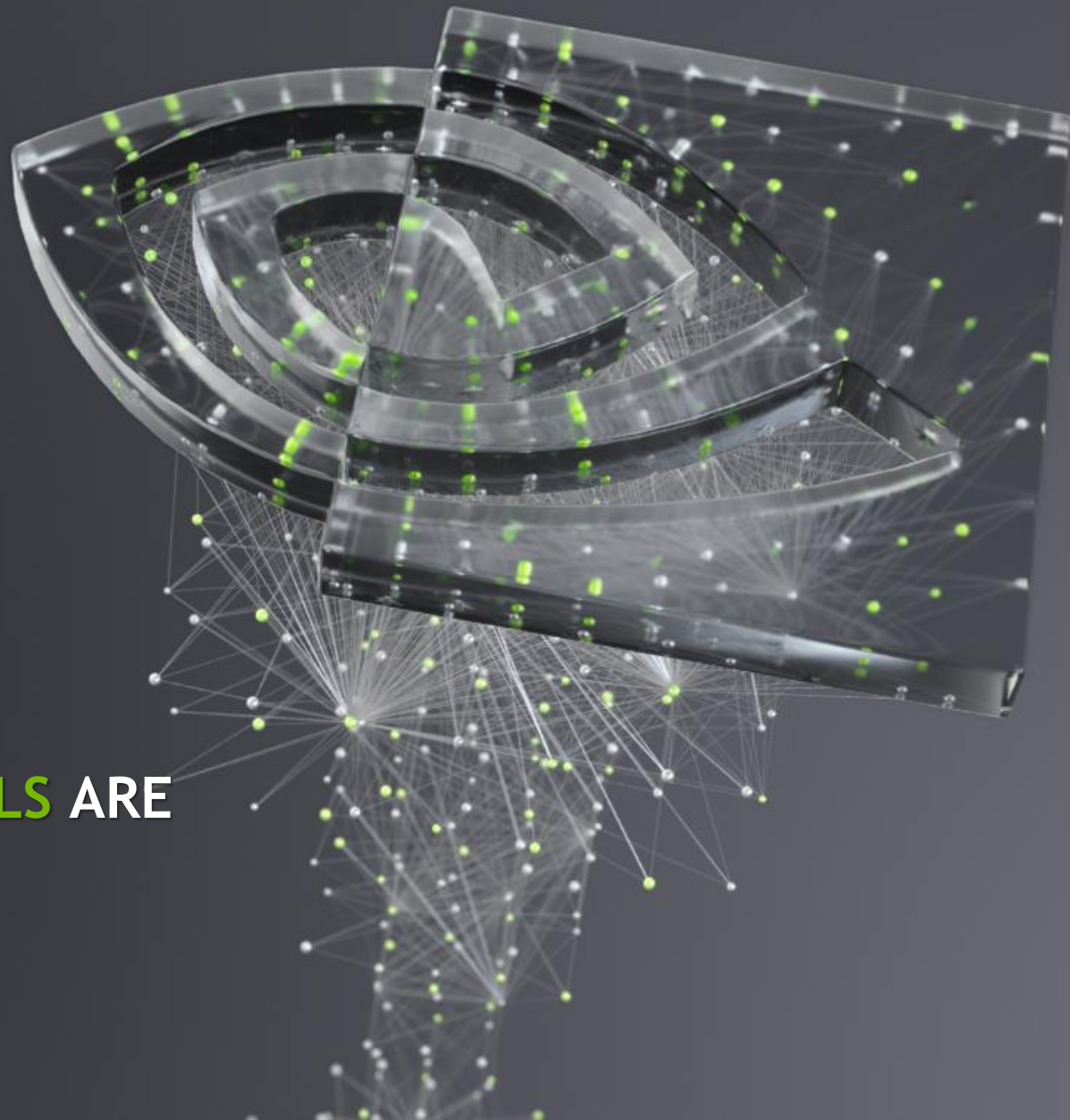




CUDA IS EVOLVING: THE LATEST **DEVELOPER TOOLS** ARE ADAPTING TO KEEP UP

Jackson Marusarz - April 12, 2021



I'M AVAILABLE TO CHAT DURING THIS SESSION

The screenshot displays the NVIDIA GTC 2021 live stream interface. The main video area shows NVIDIA CEO Jensen Huang speaking. Below the video, the title "GTC 2021 Opening Keynote with NVIDIA CEO Jensen Huang" is visible, along with a "LIVE" indicator. The sidebar on the right contains a list of sessions and a chat section. A red box highlights the "ASK THE PRESENTER / MODERATOR" button in the chat section. Another red box highlights the "1:1 CHAT" button at the bottom of the sidebar.

NVIDIA GTC 21

Search

Kevin Berry
12 Connected | 1 Requests | 3 Pending

NVIDIA GTC 21 NVIDIA GTC 2021 Opening Keynote [Session]
Happy you could join us

ASK THE PRESENTER / MODERATOR

NVIDIA GTC 21 NVIDIA GTC 2021 [Session]
Join now to the opening keynote · 16:00 →

NVIDIA GTC 21 Hands-on Open code demo [Session]
Join now to the opening keynote · 15:00 →

BD Brian Denis
Can you send me the VOD? · 14:45 →

DW Donald Watkins
Should be on time · 14:24 →

ES Einnate Schwartz
Tnx! · 12:47 →

EF Ella Farrow
See you there! · 12:45 →

JF Jessica Fisher
Hi! · 12:22 →

AB Addie Bradley
I'll ask my director [EXPLORE] →

Frank Padilla
[GTC] Keynote Oct 2021
1: "The Coming Age" [1:1 CHAT]

GTC 2021 Opening Keynote with NVIDIA CEO Jensen Huang
Appears in **NVIDIA GTC**

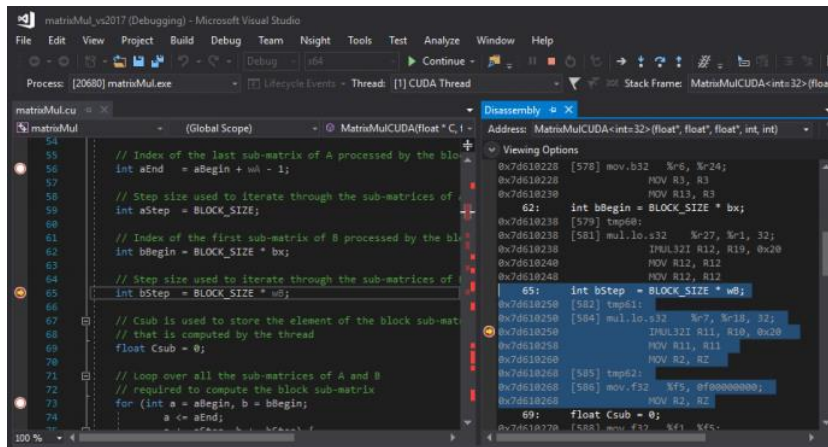
Details **Attachments**

NVIDIA CEO Jensen Huang's keynote address at the GPU Technology Conference 2021 in Silicon Valley, where he introduced breakthroughs in pro graphics with NVIDIA Omniverse; in data science with NVIDIA-powered Data Science Workstations; in inference and enterprise computing with NVIDIA T4 GPU-powered servers; in autonomous machines with NVIDIA Jetson Nano and the NVIDIA Isaac SDK; in autonomous vehicles with NVIDIA Safety Force Field and DRIVE Constellation; and much more.

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



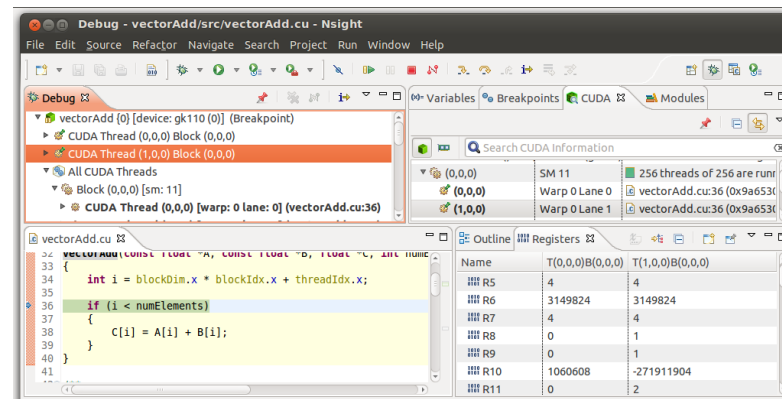
Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)



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





COMPUTE DEBUGGERS

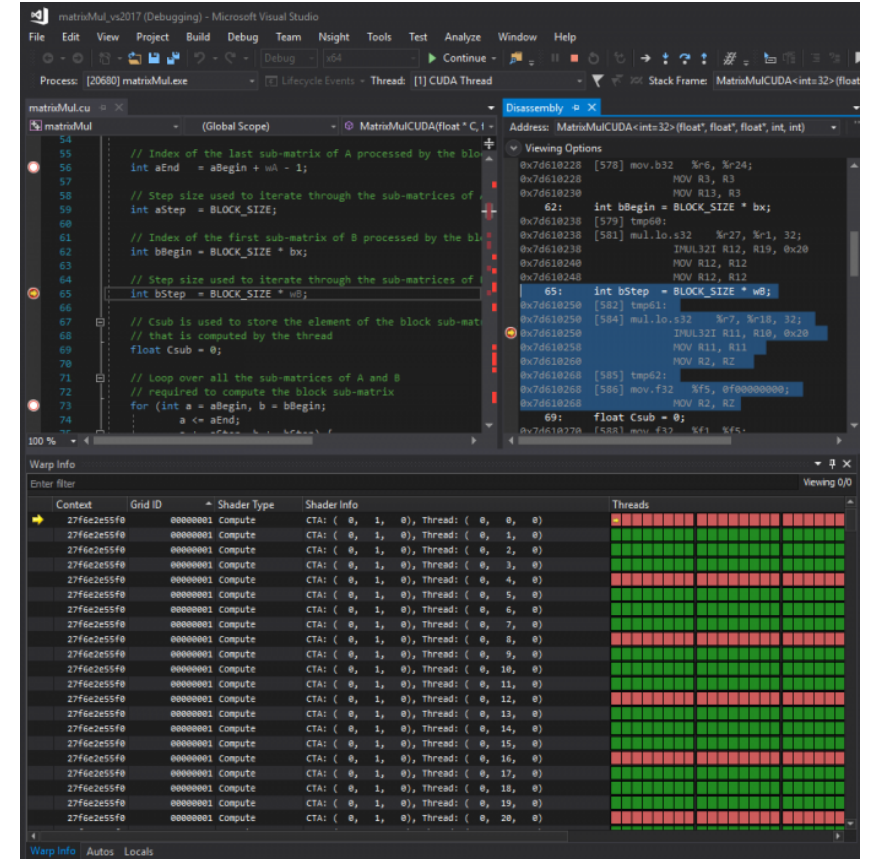
NEW FEATURES

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'

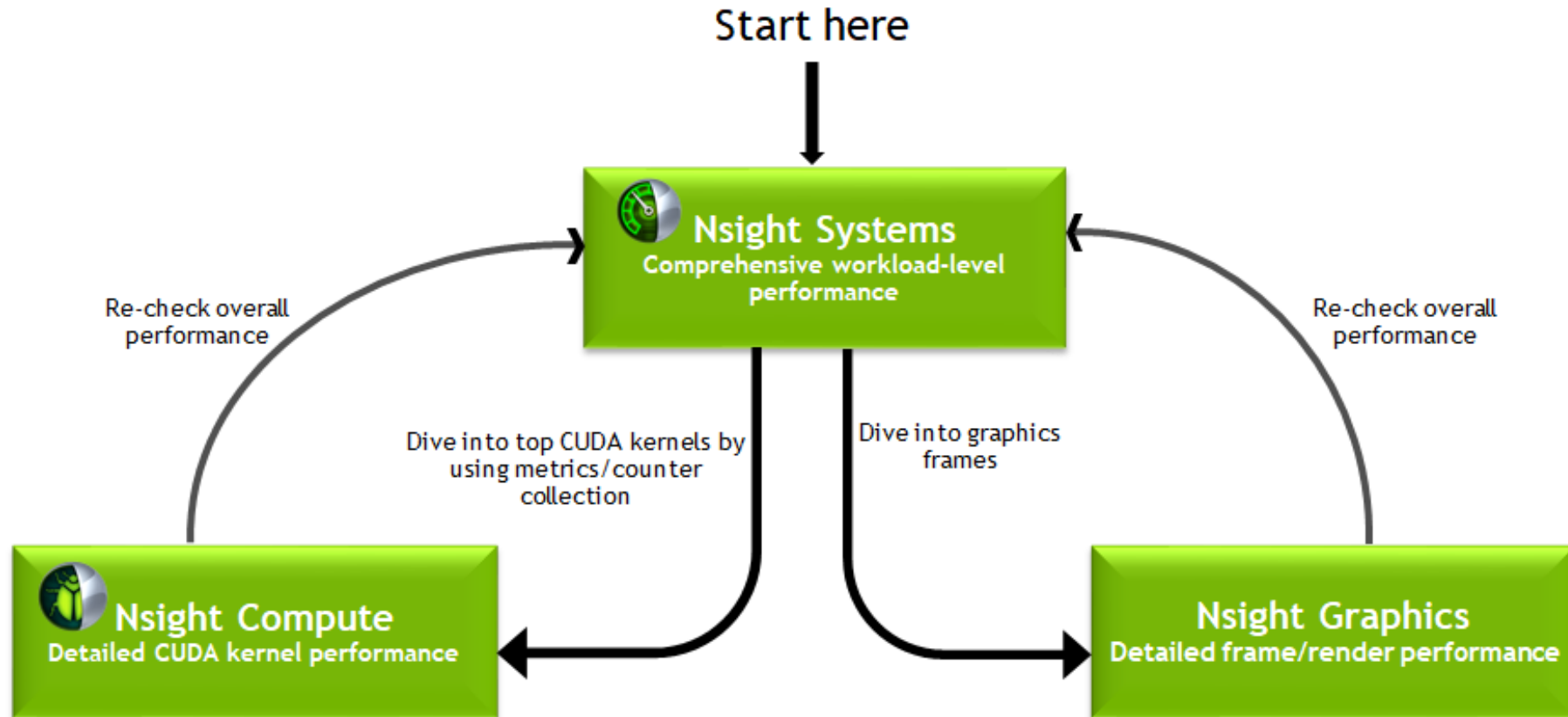
```
~/W/n/c/build $ cmake .. && cmake --build ..
-- Configuring done
-- Generating done
-- Build files have been written to: /home/rmaynard/Work/misc/cuda_sanitizer_ctest/build
[2/2] Linking CUDA executable demo
~/W/n/c/build $ ctest -D MemoryCheck
Site: RMAYNARD-DT
Build name: Linux-unknown
Create new tag: 20210325-1346 - Experimental
Configure project
  Each . represents 1024 bytes of output
  . Size of output: 0K
Build project
  Each symbol represents 1024 bytes of output.
  '!' represents an error and '*' a warning.
  . Size of output: 0K
  0 Compiler errors
  0 Compiler warnings
Performing coverage
Cannot find any coverage files. Ignoring Coverage request.
Memory check project /home/rmaynard/Work/misc/cuda_sanitizer_ctest/build
Start 1: verify
1/1 MemCheck #1: verify ..... Passed 6.77 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) = 6.77 sec
-- Processing memory checking output:
1/1 MemCheck: #1: verify ..... Defects: 4
MemCheck log files can be found here: (<#> corresponds to test number)
/home/rmaynard/Work/misc/cuda_sanitizer_ctest/build/Testing/Temporary/MemoryChecker.<#>.log
Memory checking results:
Invalid __global__ read - 1
cudaErrorLaunchFailure - 3
Submit files
SubmitURL: http://my.cdash.org/submit.php?project=CMakeTutorial
Uploaded: /home/rmaynard/Work/misc/cuda_sanitizer_ctest/build/Testing/20210325-1346/Config
Uploaded: /home/rmaynard/Work/misc/cuda_sanitizer_ctest/build/Testing/20210325-1346/Build
Uploaded: /home/rmaynard/Work/misc/cuda_sanitizer_ctest/build/Testing/20210325-1346/Dynam
Uploaded: /home/rmaynard/Work/misc/cuda_sanitizer_ctest/build/Testing/20210325-1346/Done.
Submission successful
~/W/n/c/build $
```

NSIGHT PROFILING TOOLS WORKFLOW

ITERATIVE OPTIMIZATION FOR COMPUTE AND GRAPHICS



An abstract graphic featuring a complex network of thin, glowing green lines that crisscross the frame. These lines connect various points, some of which are highlighted as bright green nodes. The background is a deep, dark blue or black, with some larger, faint, circular bokeh-like shapes in shades of blue and green. The overall effect is one of a dynamic, interconnected system, possibly representing a network or data flow.

NSIGHT SYSTEMS



NSIGHT SYSTEMS

SYSTEM PROFILER

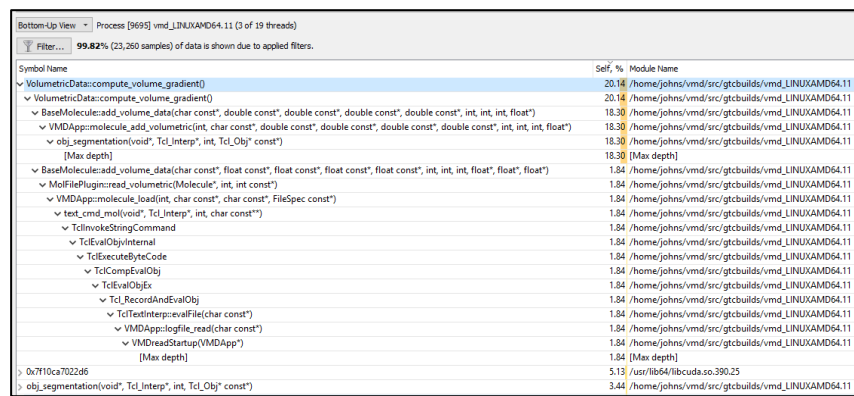
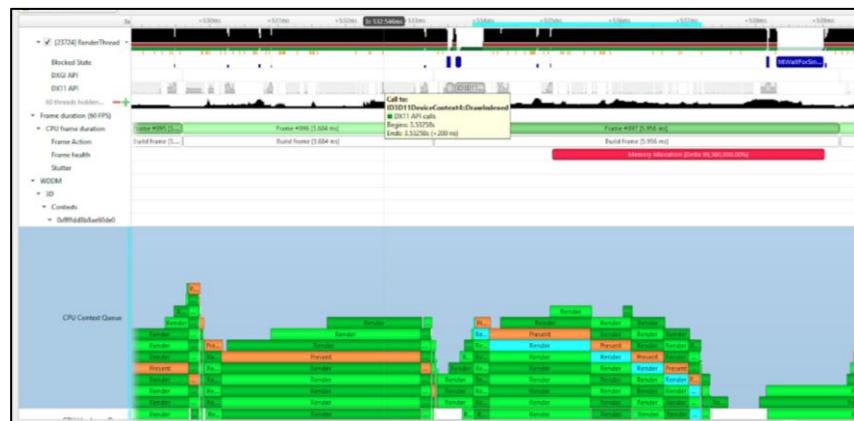
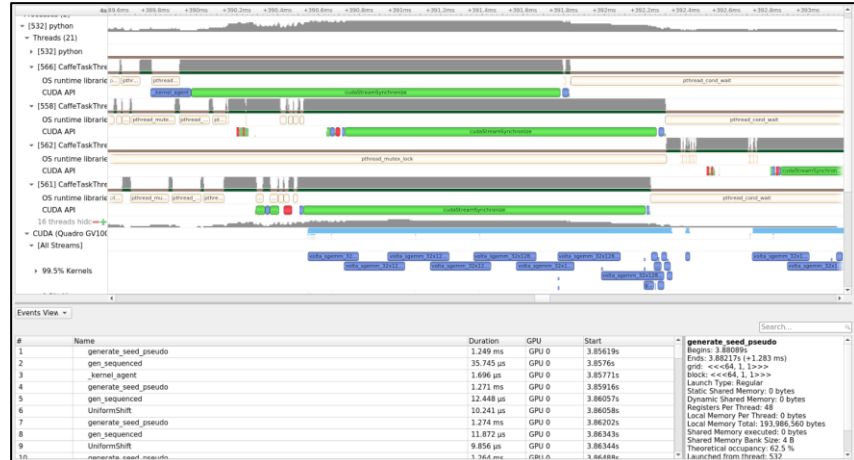
Key Features:

- 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>



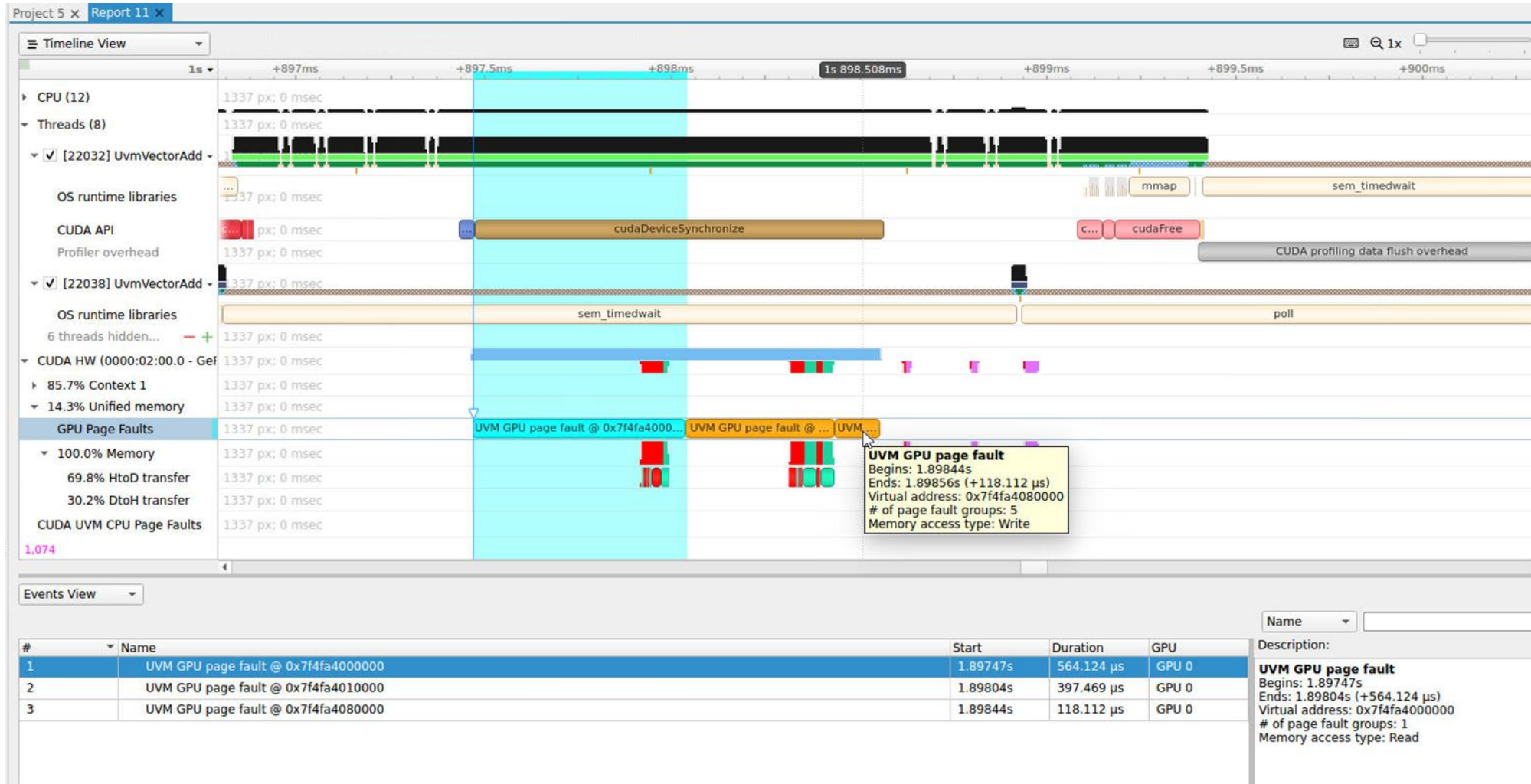


NSIGHT SYSTEMS

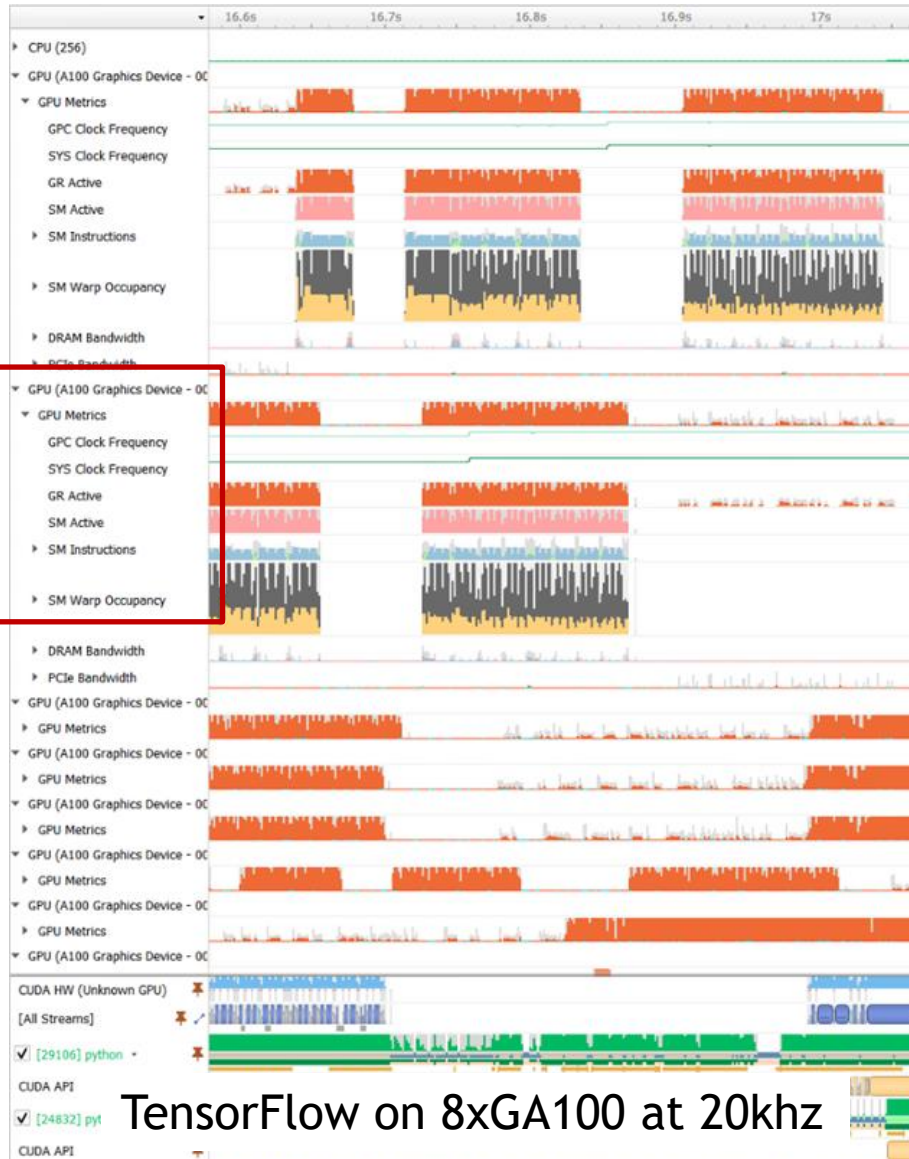
NEW FEATURES

CUDA UNIFIED MEMORY CPU & GPU PAGE FAULT TRACE

FIND PERFORMANCE IMPACTS FROM PAGING



GPU METRICS SAMPLING



- 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

An abstract graphic featuring a network of glowing green nodes connected by thin, intersecting lines. The nodes are scattered across the frame, with some appearing as bright points and others as larger, softer glows. The lines create a complex web of connections, suggesting a digital or computational theme. The background is a deep, dark blue or black, which makes the green elements stand out.

NSIGHT COMPUTE



NSIGHT COMPUTE

KERNEL PROFILING TOOL

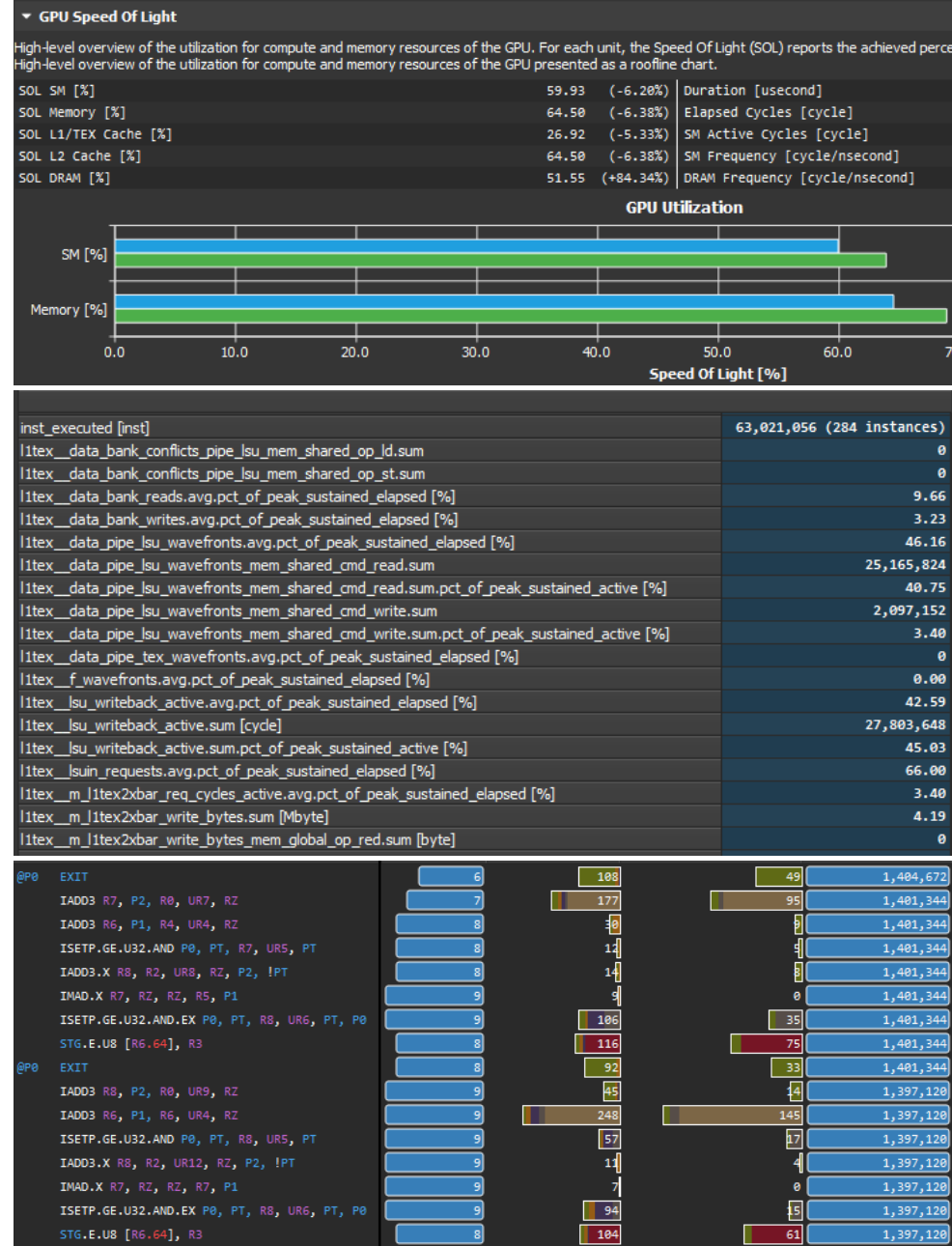
Key Features:

- 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>



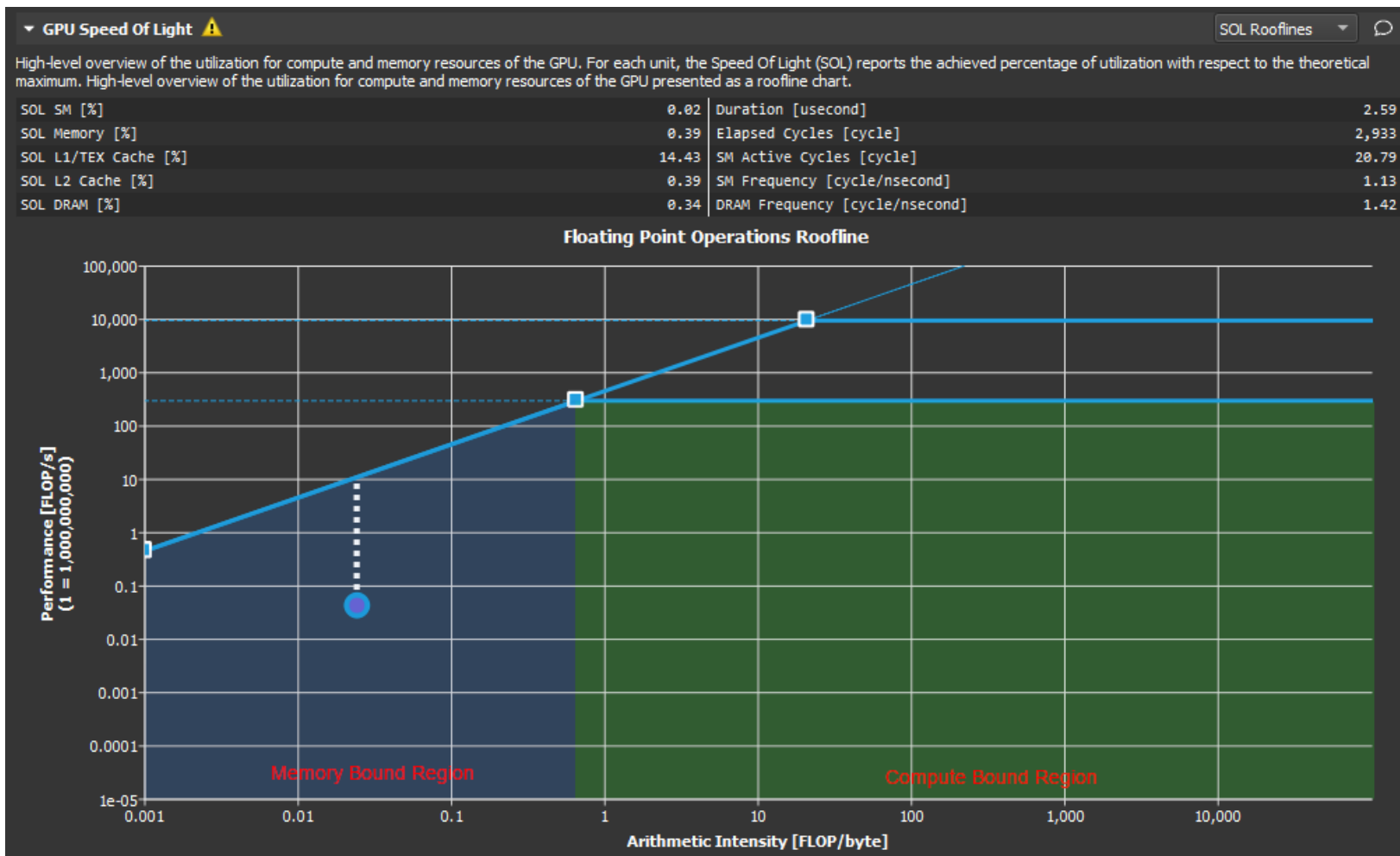


NSIGHT COMPUTE

NEW FEATURES

ROOFLINE ANALYSIS

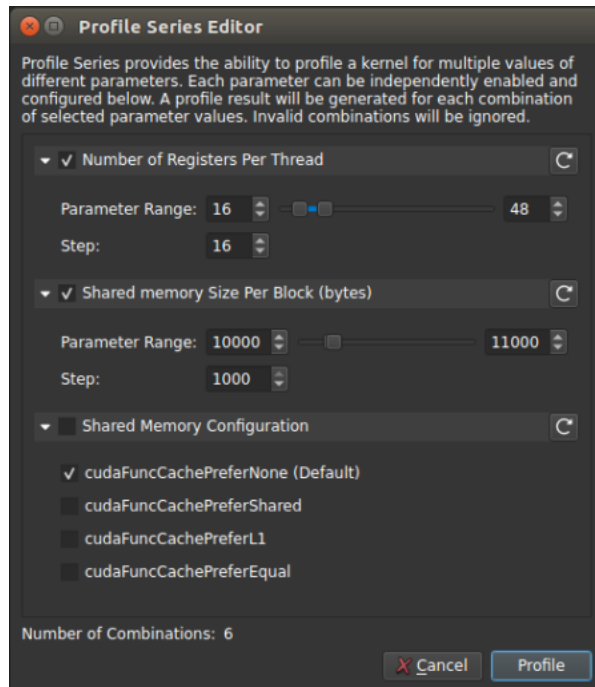
FIND HARDWARE LIMITATIONS FOR CUDA KERNELS



- 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

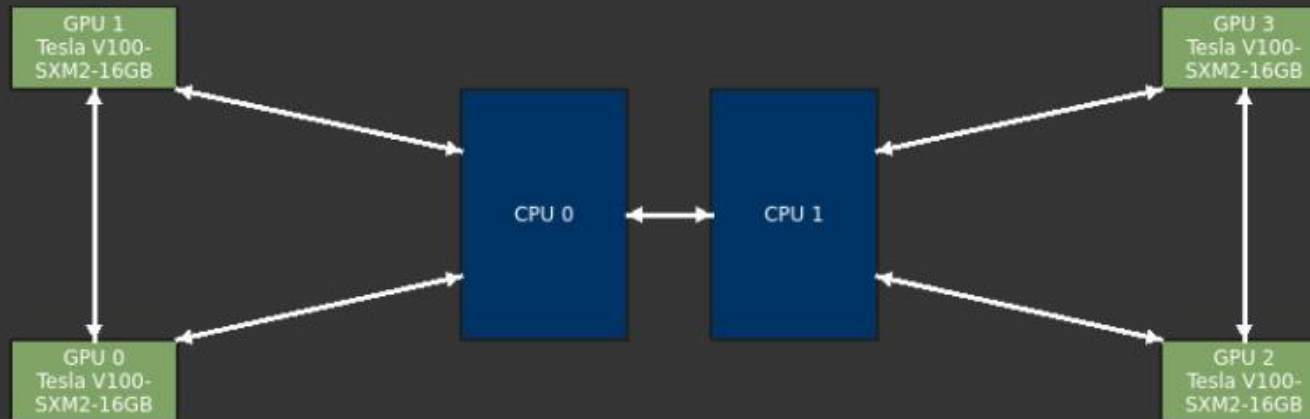
Device-level
Metrics

▼ NVLink

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)

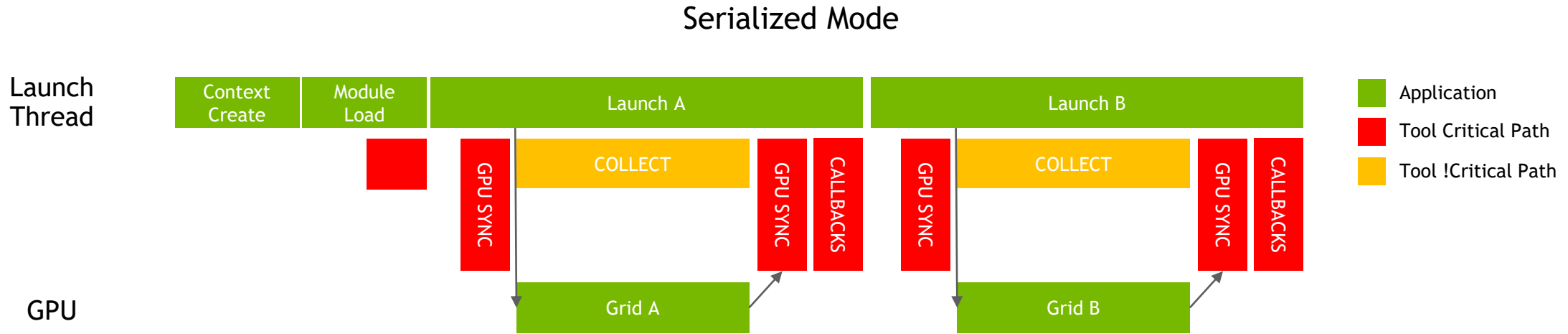
- Collects GPU Program Counter (PC) and Stall reason (if applicable)
- Sampling modes - Continuous (concurrent kernels) and Serialized (sequential kernels)
 - Performance Numbers for continuous mode on Volta GPU:
 - Single GPU: <2% for sampling period 11 (2^{11} cycles) on CuLaunches, raja-perf, GROMACS
 - 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

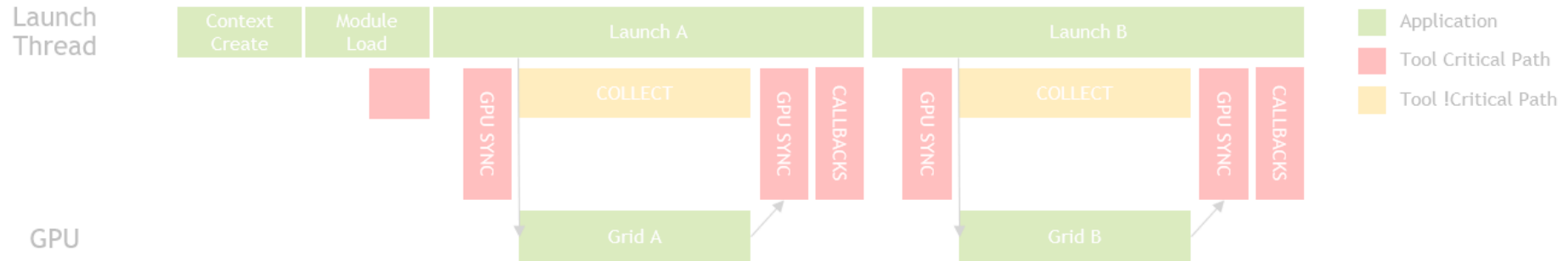
SERIALIZED KERNELS LIMIT PERFORMANCE



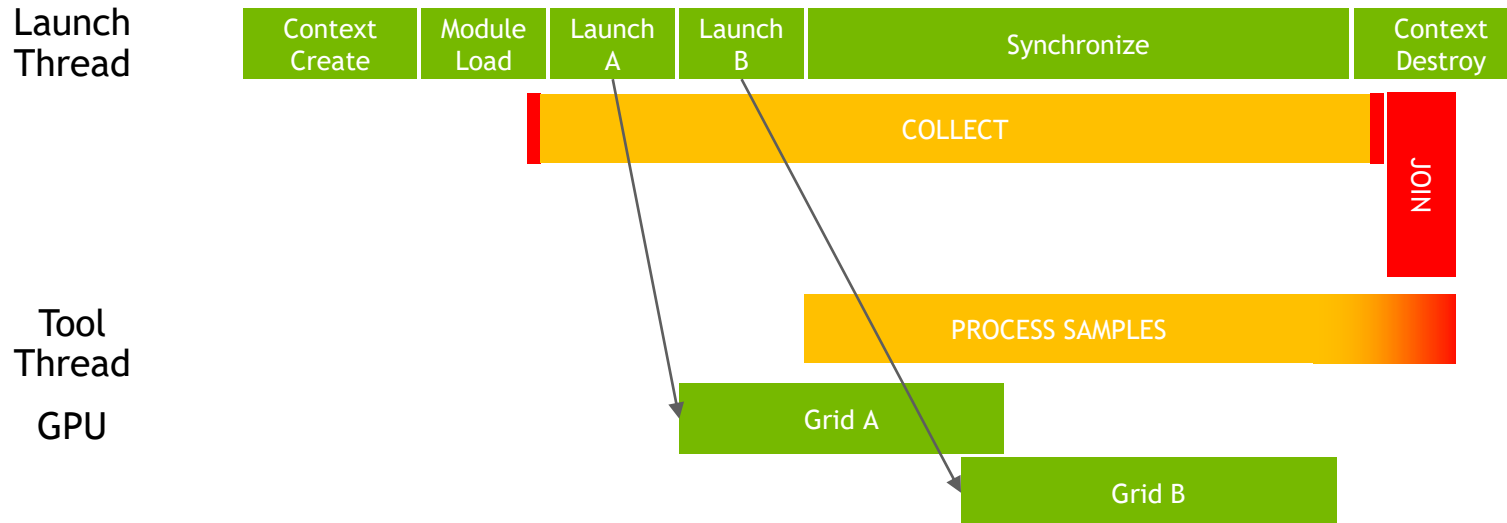
PC SAMPLING MODES

CONTINUOUS SAMPLING FOR BETTER REALISTIC PERFORMANCE

Serialized Mode



Continuous Mode



The background is a dark, almost black, field with a complex network of thin, glowing green lines. These lines intersect at various points, creating a web-like structure. At many of these intersection points, there are small, bright green dots or nodes. Some of these dots are slightly larger and more intense than others. The overall effect is reminiscent of a neural network diagram, a data visualization, or a stylized representation of a complex system. The lines and dots are scattered across the upper two-thirds of the image, leaving the lower third mostly clear for the text.

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:

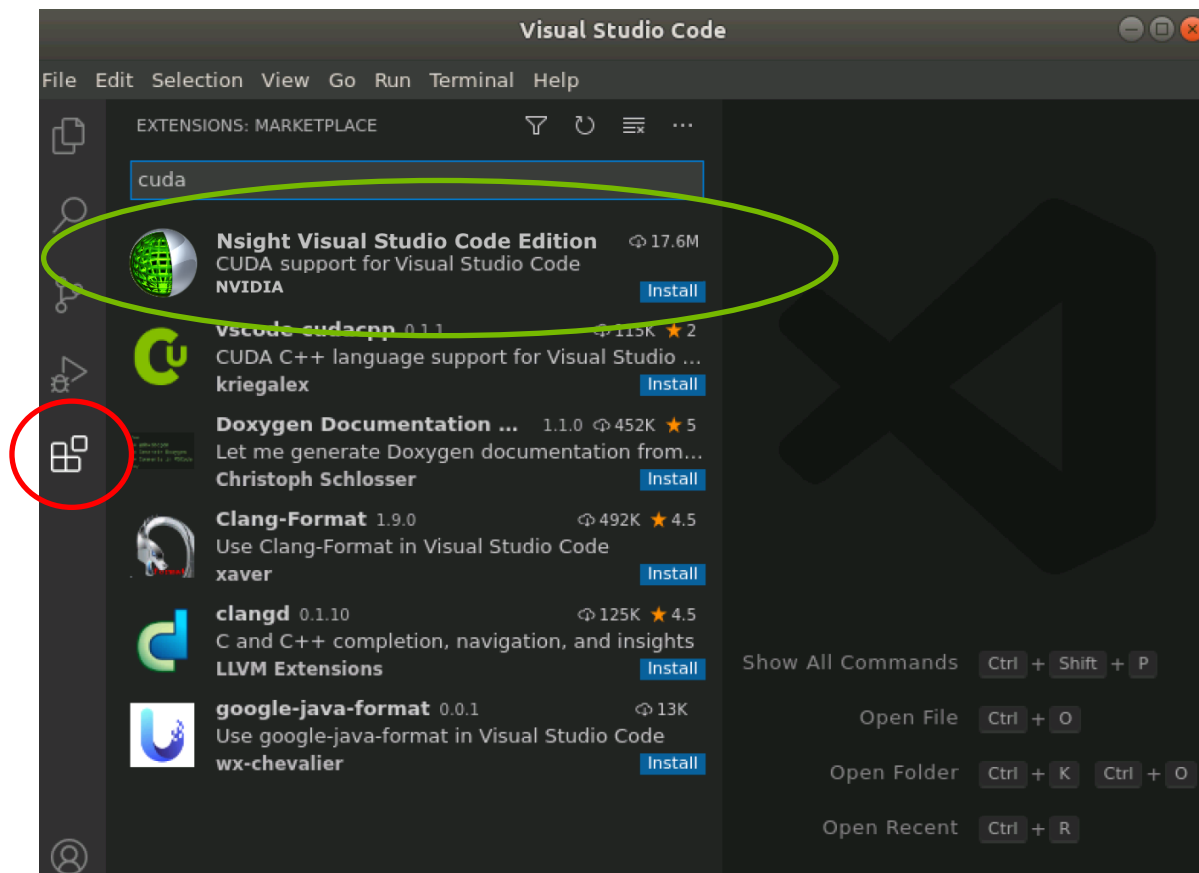
- CUDA code syntax highlighting
- CUDA code completion
- Build warning/errors
- Debug CPU & GPU code

COMING SOON

<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

The screenshot displays the Visual Studio Code interface for debugging a CUDA program. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar reads "matrixMul.cu - matrixMul - Visual Studio Code".

VARIABLES Panel: Shows local variables for the current thread. Key variables include:

- Local:**
 - Bs: 0x1000
 - As: 0x0
 - C: 0x7fffce52c000
 - A: 0x7fffce400000
 - *A: 1
 - B: 0x7fffce464000
 - wA: 320
 - wB: 640
 - a: 0
 - b: 32
 - by: 0
 - tx: 0
 - aStep: 32
 - bx: 1
 - ty: 28
 - aBegin: 0
 - aEnd: 319
 - bBegin: 32
 - bStep: 20480
 - Csub: 0
 - c: <optimized out>
- Registers:**
 - R0: 0x0000ce52c000
 - R1: 0x000000ffde0
 - R2: 0x00000007fff
 - R3: 0x0000ce400000

WATCH Panel: Displays the value of the variable `As` as `[32]`.

BREAKPOINTS Panel: Lists four breakpoints set in `matrixMul.cu` at lines 61, 64, 72, and 84.

Source Code: The central editor shows the `matrixMul.cu` file. The code is a CUDA kernel for matrix multiplication. A red dot indicates the current execution point at line 84, and a yellow highlight is on line 88 (`syncthreads();`).

DEBUG CONSOLE: Shows the output of the `-exec info cuda warps` command, displaying information about active lanes and warps.

CALL STACK: Shows the call stack with the current function being `matrixMul.cu` at line 88.

Status Bar: At the bottom, it indicates the current position: "Ln 88, Col 1" and provides details about the execution context: "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 \[T2504\]](#)

