



NVIDIA PROFILING TOOLS

KRISTOPHER KEIPERT

PROGRAMMING THE NVIDIA PLATFORM

CPU, GPU, and Network

ACCELERATED STANDARD LANGUAGES

ISO C++, ISO Fortran

```
std::transform(par, x, x+n, y, y,  
    [=](float x, float y){ return y +  
a*x; }  
);
```

```
do concurrent (i = 1:n)  
    y(i) = y(i) + a*x(i)  
enddo
```

```
import cunumeric as np  
...  
def saxpy(a, x, y):  
    y[:] += a*x
```

INCREMENTAL PORTABLE OPTIMIZATION

OpenACC, OpenMP

```
#pragma acc data copy(x,y) {  
...  
std::transform(par, x, x+n, y, y,  
    [=](float x, float y){  
        return y + a*x;  
    });  
...  
}  
  
#pragma omp target data map(x,y) {  
...  
std::transform(par, x, x+n, y, y,  
    [=](float x, float y){  
        return y + a*x;  
    });  
...  
}
```

PLATFORM SPECIALIZATION

CUDA

```
__global__  
void saxpy(int n, float a,  
    float *x, float *y) {  
    int i = blockIdx.x*blockDim.x +  
        threadIdx.x;  
    if (i < n) y[i] += a*x[i];  
}  
  
int main(void) {  
    ...  
    cudaMemcpy(d_x, x, ...);  
    cudaMemcpy(d_y, y, ...);  
  
    saxpy<<<(N+255)/256,256>>>(...);  
  
    cudaMemcpy(y, d_y, ...);  
}
```

ACCELERATION LIBRARIES

Core

Math

Communication

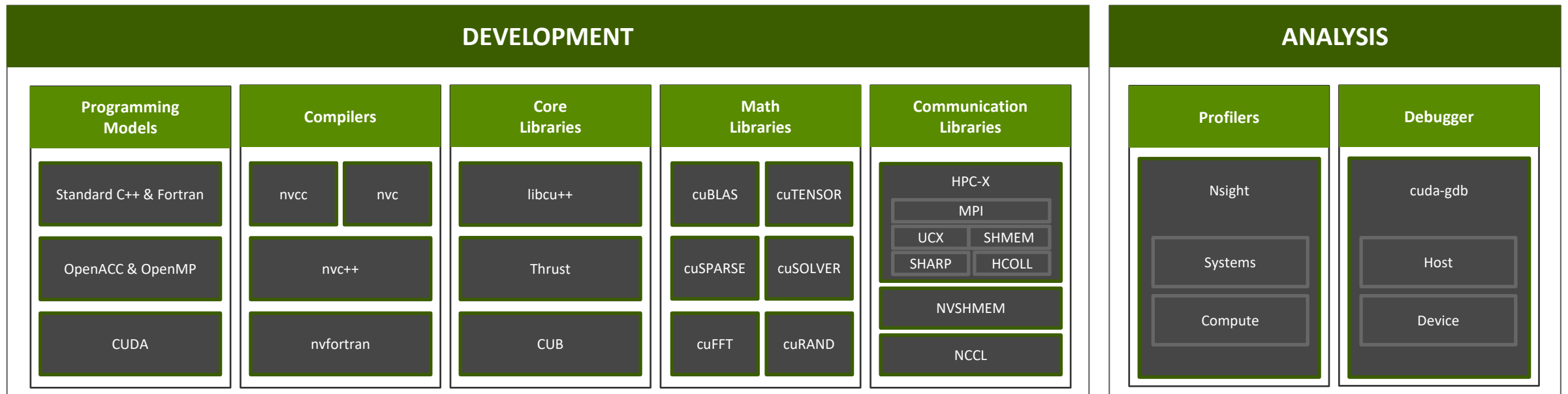
Data Analytics

AI

Quantum

NVIDIA HPC SDK

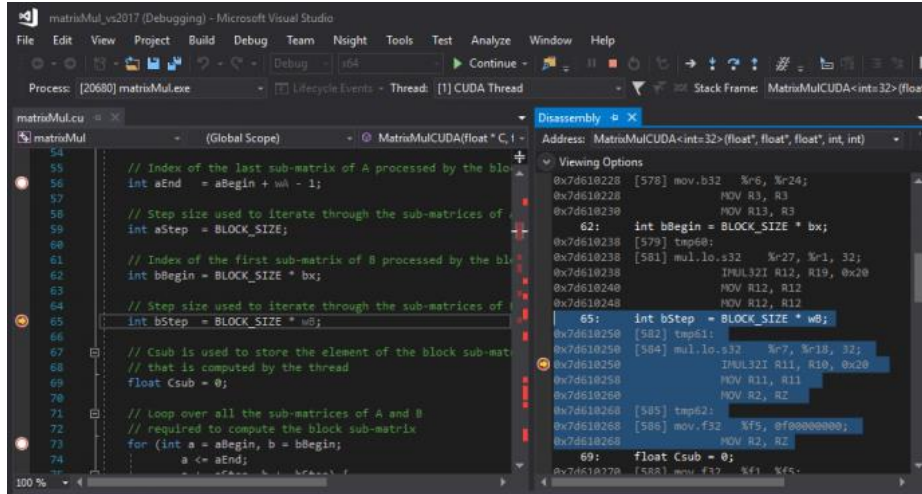
Available at developer.nvidia.com/hpc-sdk, on NGC, via Spack, and in the Cloud



Develop for the NVIDIA Platform: GPU, CPU and Interconnect
Libraries | Accelerated C++ and Fortran | Directives | CUDA
7-8 Releases Per Year | Freely Available

DEVELOPER TOOLS

Debuggers: cuda-gdb, Nsight Visual Studio Edition



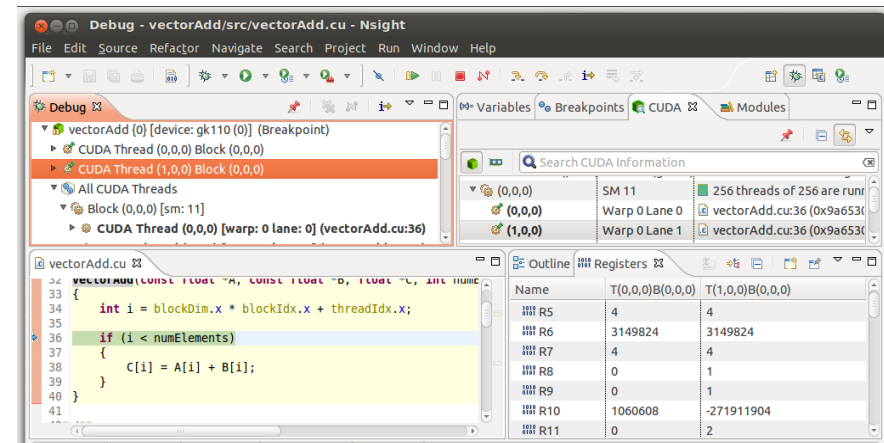
Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)



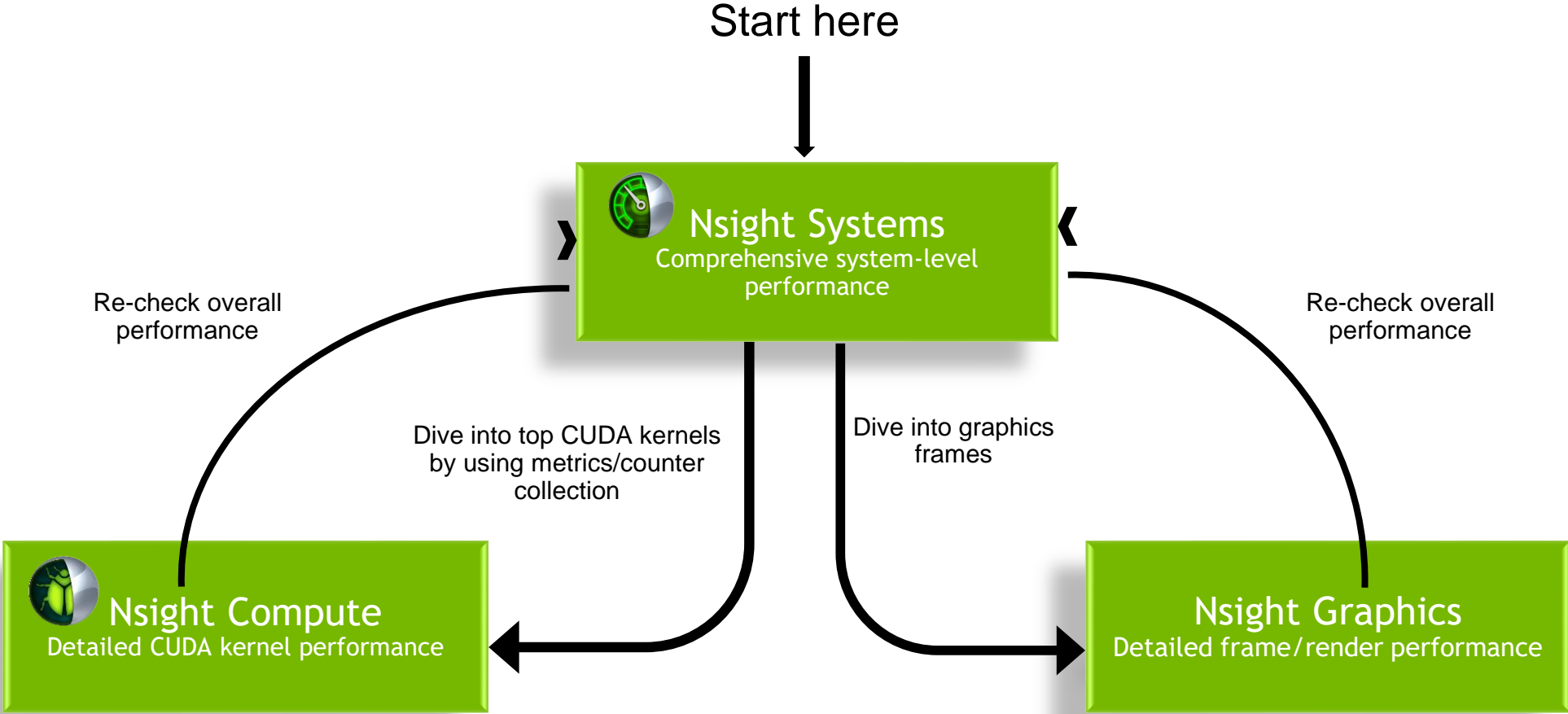
Correctness Checker::: Compute Sanitizer

```
$ 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_kernl i(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



NSIGHT TOOLS WORKFLOW





NSIGHT SYSTEMS

System Profiler

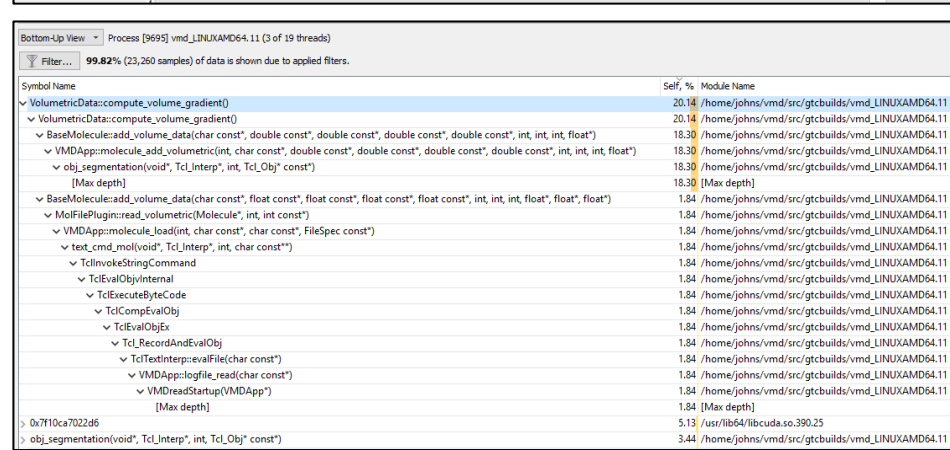
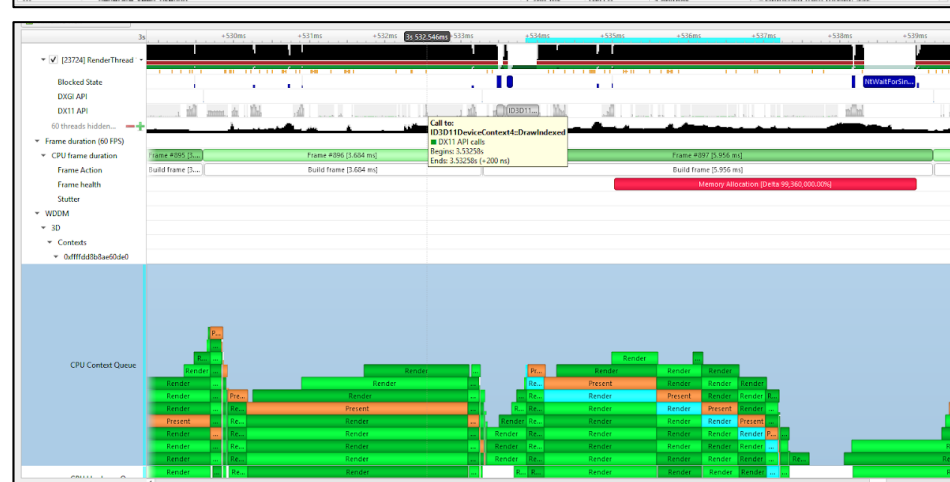
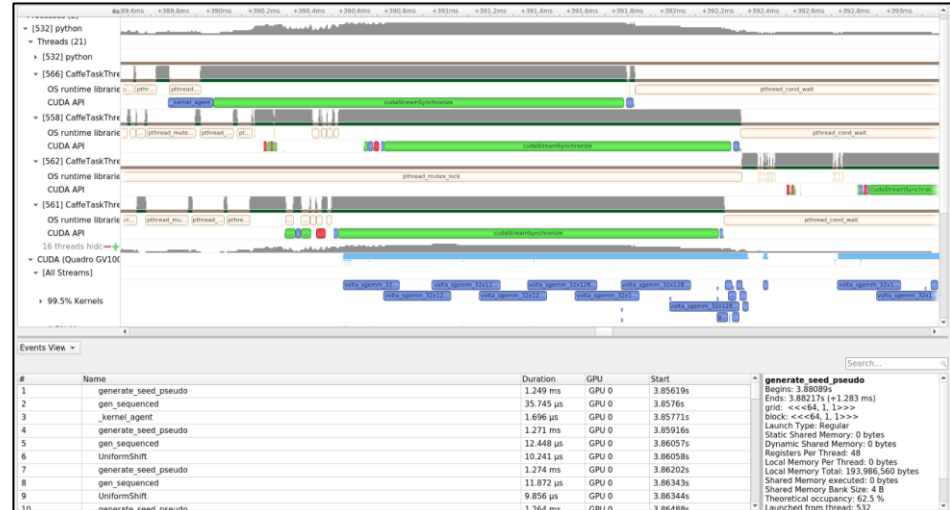
Key Features:

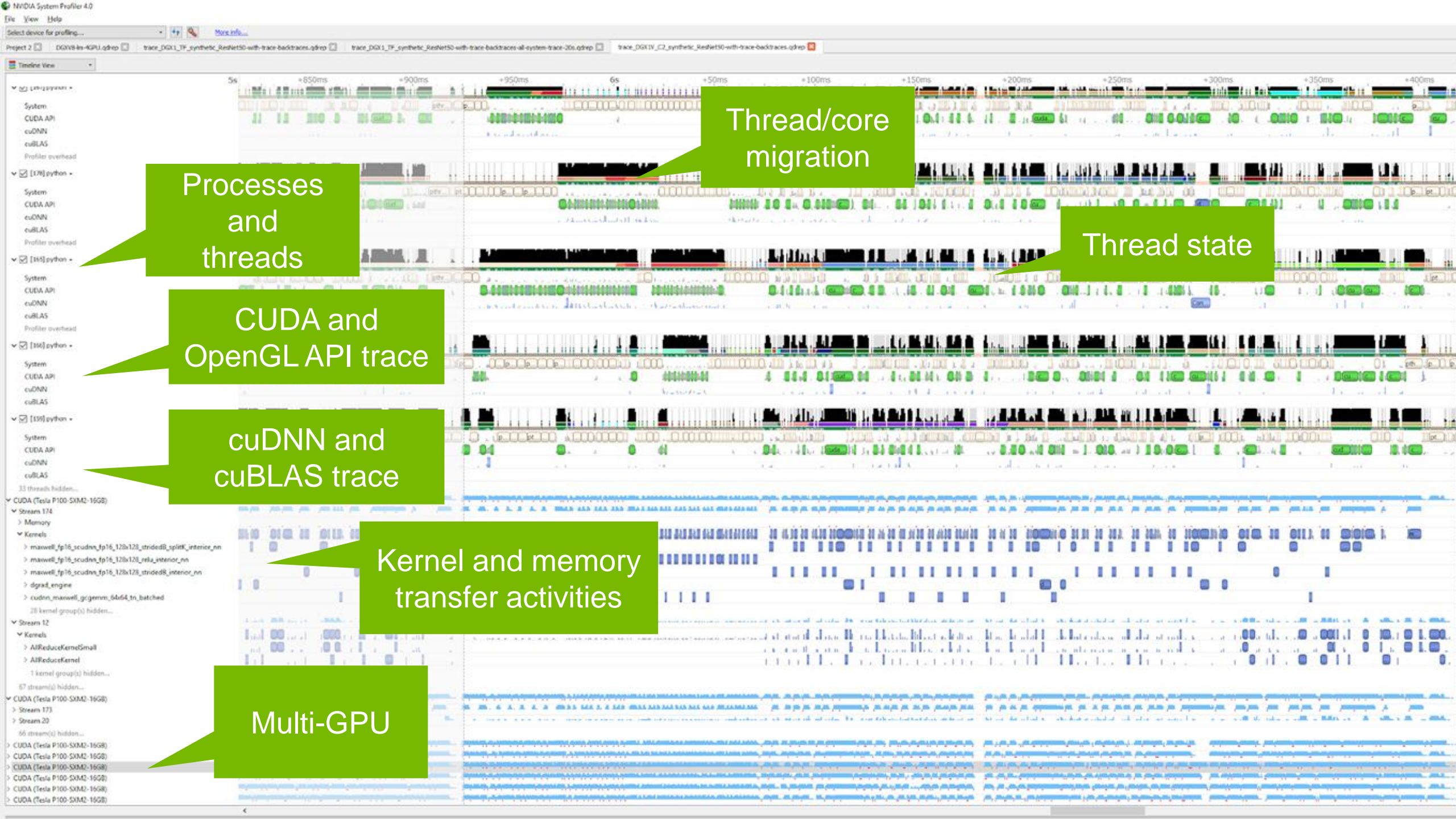
- System-wide application algorithm tuning
 - Multi-process tree support
- Locate optimization opportunities
 - Visualize millions of events on a very fast GUI timeline
 - Or gaps of unused CPU and GPU time
- Balance your workload across multiple CPUs and GPUs
 - CPU algorithms, utilization and thread 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>





Processes and threads

Thread/core migration

Thread state

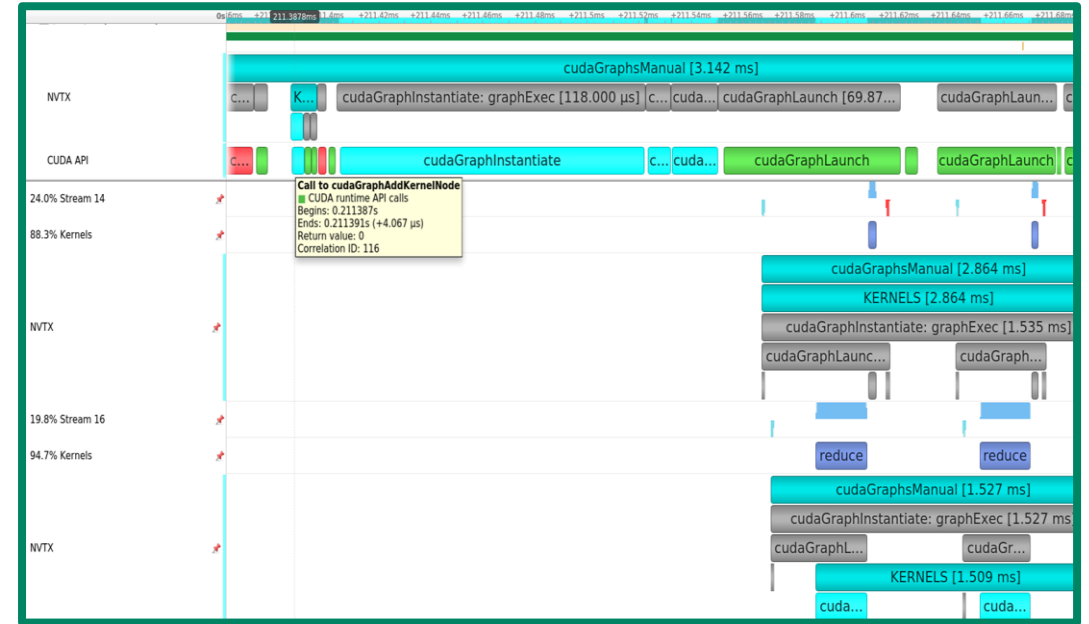
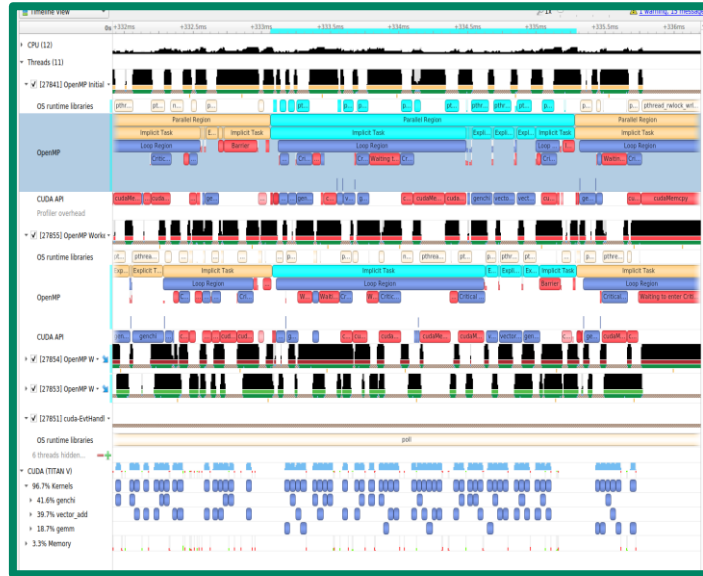
CUDA and OpenGL API trace

cuDNN and cuBLAS trace

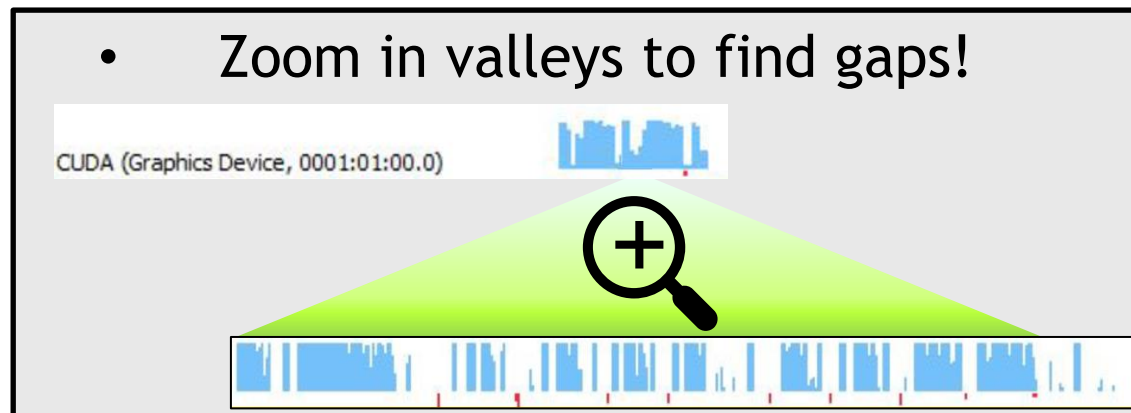
Kernel and memory transfer activities

Multi-GPU

ZOOM/FILTER TO EXACT AREAS OF INTEREST

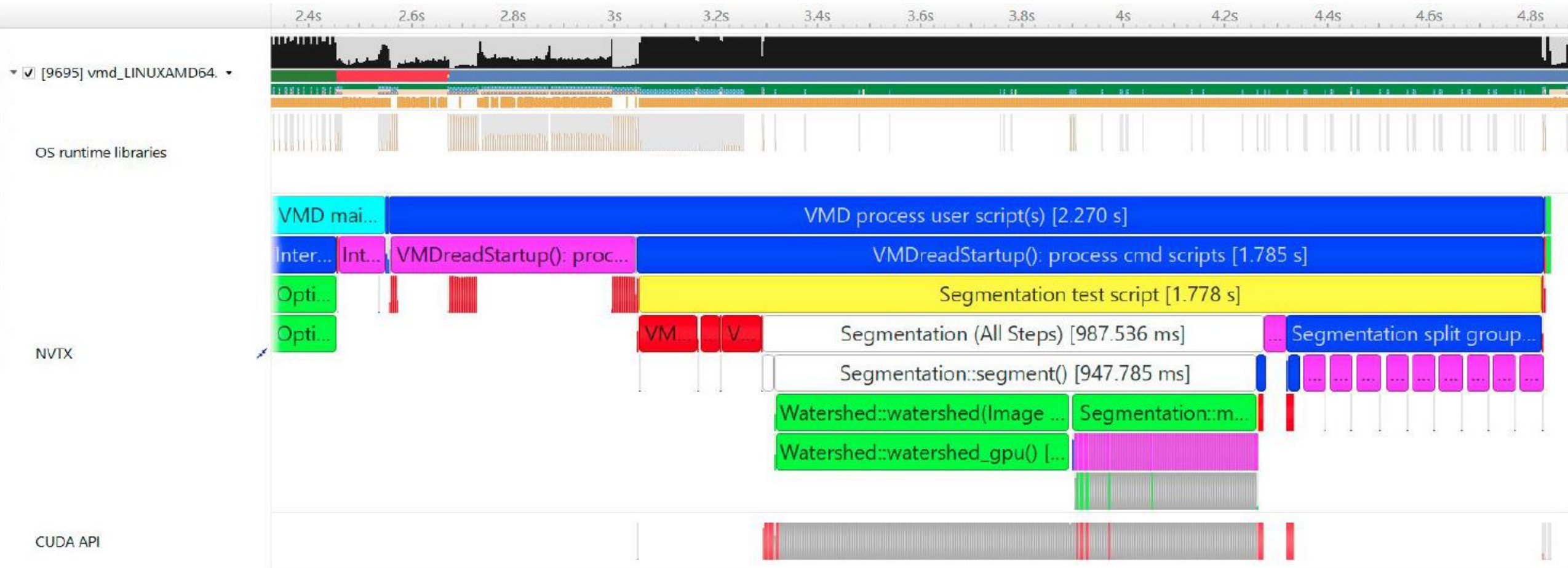


- Zoom in valleys to find gaps!



NVTX: NVIDIA TOOLS EXTENSIONS

Code Annotation API



EXPERT SYSTEMS & STATISTICS

Built-in Data Analytics with Advice

The screenshot displays the NVIDIA Nsight Systems interface. The top section shows a timeline view of a profile trace. The 'CUDA API' track highlights several 'cudaMemcpy' and 'cudaMemcpyAsync' calls. A tooltip on the right lists the following warnings:

- CUDA Async Memcpy with Pageable Memory
- CUDA Synchronous Memcpy
- CUDA Synchronous Memset
- CUDA Synchronization APIs
- CUDA GPU Starvation
- CUDA GPU Low Utilization
- VULKAN GPU Starvation
- VULKAN GPU Low Utilization

The bottom section shows the 'Expert System View' for the selected 'CUDA Async Memcpy with Pageable Memory' event. It includes a table with the following data:

Duration	Start	Src Kind	Dst Kind	Bytes	PID	Device ID	Context ID	Stream ID	API Name
2,048 μs	6,38792s	Device	Pageable	8 B	75475		0	1	7 cudaMemcpy
2,048 μs	6,8334s	Device	Pageable	4 B	75475		0	1	7 cudaMemcpy
2,016 μs	2,5394s	Device	Pageable	4 B	75475		0	1	7 cudaMemcpy
2,016 μs	3,90617s	Device	Pageable	48 B	75475		0	1	7 cudaMemcpy
2,016 μs	4,25257s	Device	Pageable	4 B	75475		0	1	7 cudaMemcpy
2,016 μs	5,67617s	Device	Pageable	48 B	75475		0	1	7 cudaMemcpy
2,016 μs	5,9572s	Device	Pageable	8 B	75475		0	1	7 cudaMemcpy
2,016 μs	5,97088s	Device	Pageable	4 B	75475		0	1	7 cudaMemcpy

Below the table, the expert system provides the following text:

The following APIs use PAGEABLE memory which causes asynchronous CUDA memcpy operations to block and be executed synchronously. This leads to low GPU utilization.

Suggestion: If applicable, use PINNED memory instead.

CLI command:
nsys analyze -r cuda-async-memcpy /mnt/data/traces/qdrep/nccl/profile_circe-n011_506451_0.sqlite

MULTI-REPORT TILING

Visualize More Parallel Activity

Open multiple reports

Loaded on same timeline based on wall-clock



APPLICATION PROFILES WITH NSIGHT SYSTEMS

```
$ nsys profile -o report --stats=true ./myapp.exe
```

- Generated file: report.qdrep (or report.nsys-rep)
Open for viewing in the Nsight Systems UI

- When using MPI, recommended to use *nsys* after *mpirun/srun*:

```
$ mpirun -n 4 nsys profile ./myapp.exe
```

PROFILING DL MODELS

- Pytorch

- DNN Layer annotations are disabled by default
- ++ *”with torch.autograd.profiler.emit_nvtx():”*
- Manually with *torch.cuda.nvtx.range_(push/pop)*
- TensorRT backend is already annotated

- Tensorflow

- Annotated by default with NVTX in NVIDIA TF containers
- `TF_DISABLE_NVTX_RANGES=1` to disable for production



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)

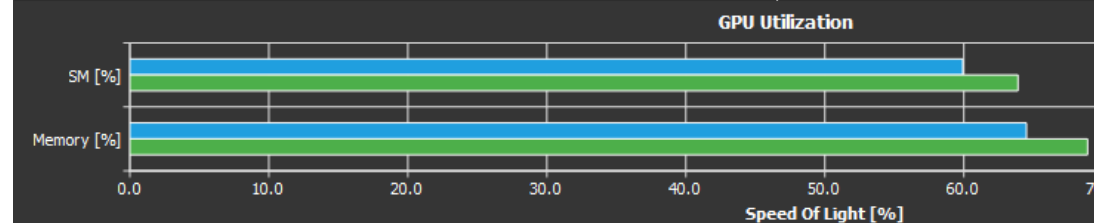
GPUs: Volta+

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

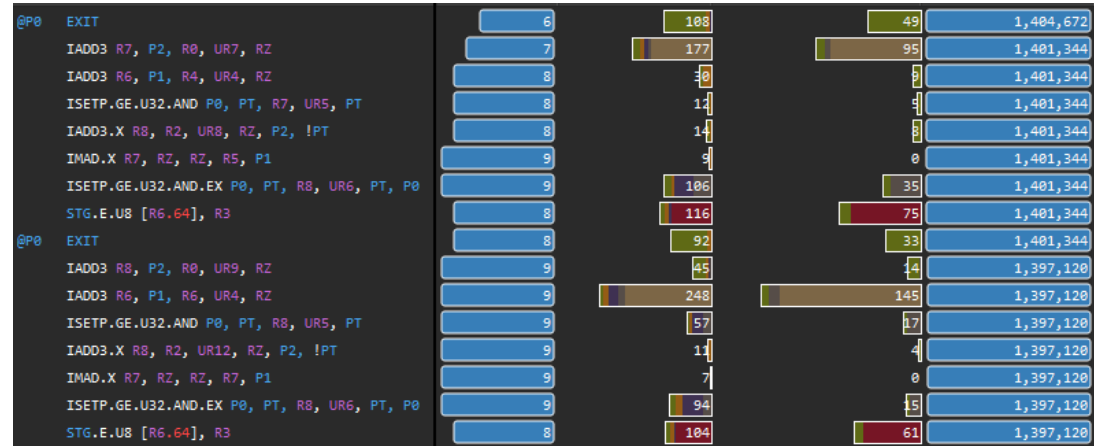
GPU Speed Of Light

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

SOL SM [%]	59.93 (-6.20%)	Duration [usecond]
SOL Memory [%]	64.50 (-6.38%)	Elapsed Cycles [cycle]
SOL L1/TEX Cache [%]	26.92 (-5.33%)	SM Active Cycles [cycle]
SOL L2 Cache [%]	64.50 (-6.38%)	SM Frequency [cycle/nsecond]
SOL DRAM [%]	51.55 (+84.34%)	DRAM Frequency [cycle/nsecond]



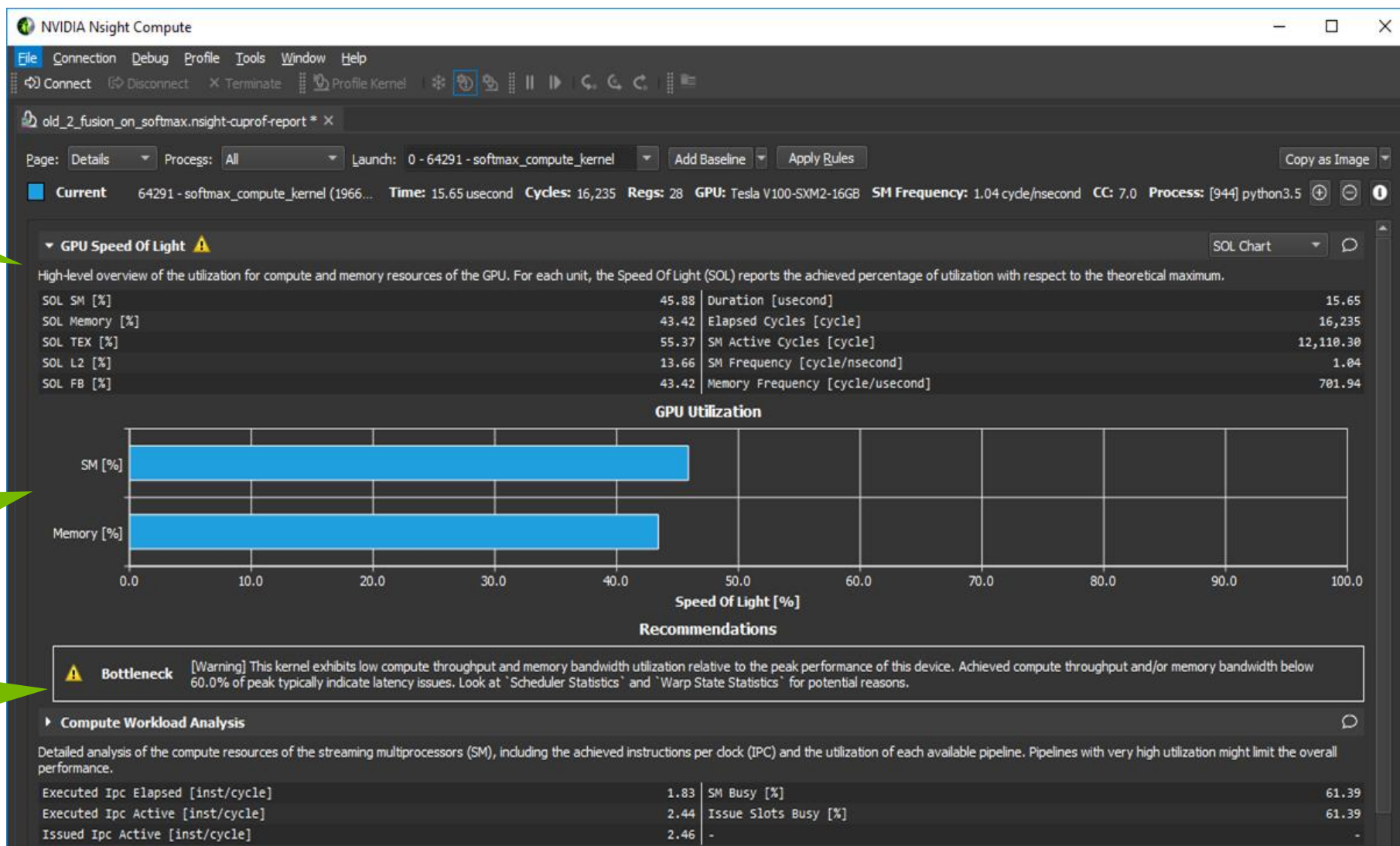
inst_executed [inst]	63,021,056 (284 instances)
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.sum	0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.sum	0
l1tex__data_bank_reads.avg.pct_of_peak_sustained_elapsed [%]	9.66
l1tex__data_bank_writes.avg.pct_of_peak_sustained_elapsed [%]	3.23
l1tex__data_pipe_lsu_wavefronts.avg.pct_of_peak_sustained_elapsed [%]	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



Targeted metric sections

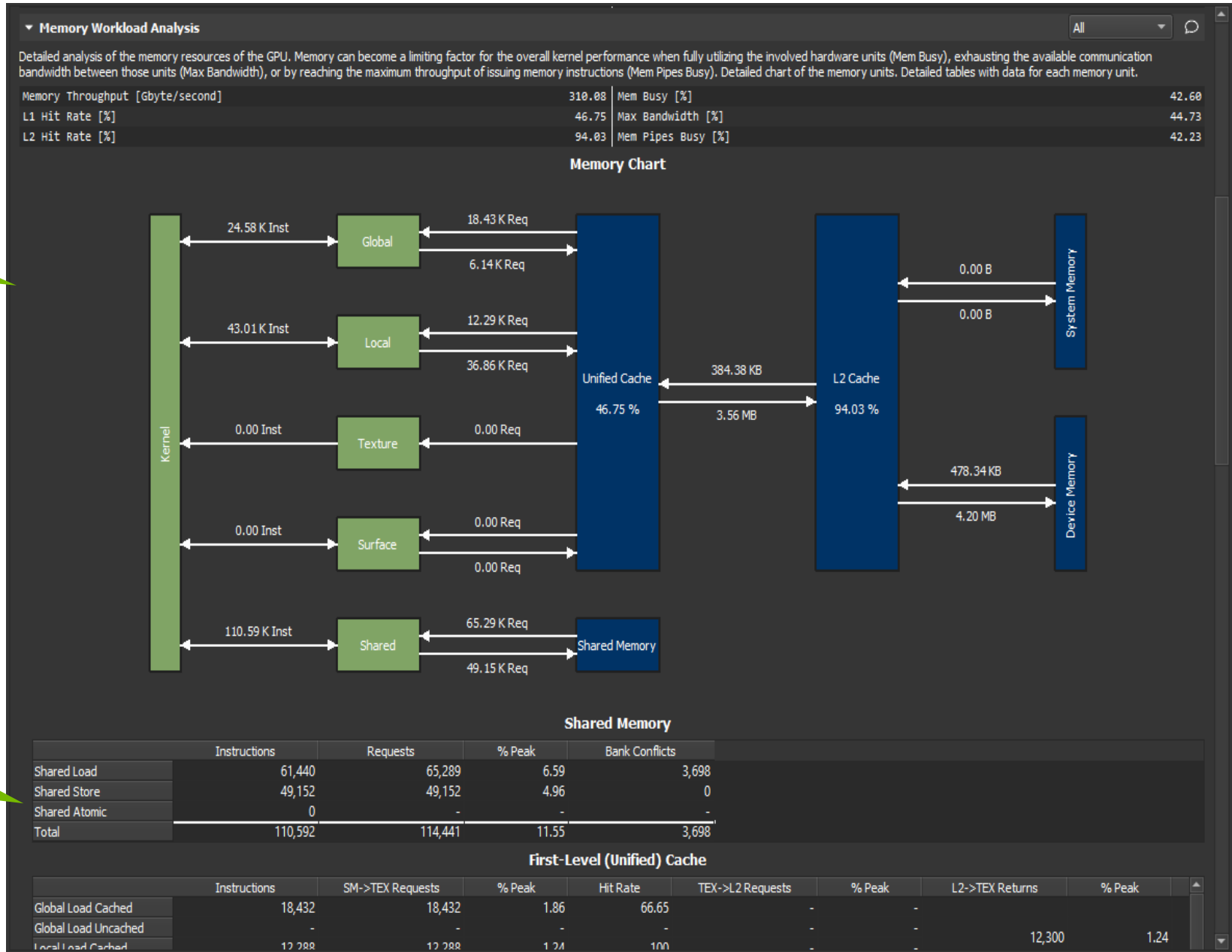
Customizable data collection and presentation

Built-in expertise for Guided Analysis and optimization



Visual memory analysis chart

Metrics for peak performance ratios



The screenshot displays the NVIDIA Nsight Systems interface, showing the analysis of a softmax kernel. The interface is divided into several panels:

- Source Code (Left):** Shows the C++ source code for the `softmax_compute_kernel` function. A green callout points to this section with the text: "Source/PTX/SASS analysis and correlation".
- Sampling Data (Middle):** A table showing the execution of instructions. A green callout points to this table with the text: "Metric heatmap to quickly identify hotspots".
- PTX/SASS (Right):** Shows the corresponding PTX/SASS instructions for the source code.
- Sampling Data (All) (Center):** A summary table showing the total sample count and various metrics for each instruction. A green callout points to this table with the text: "Source metrics per instruction".

Sampling Data (All) Summary:

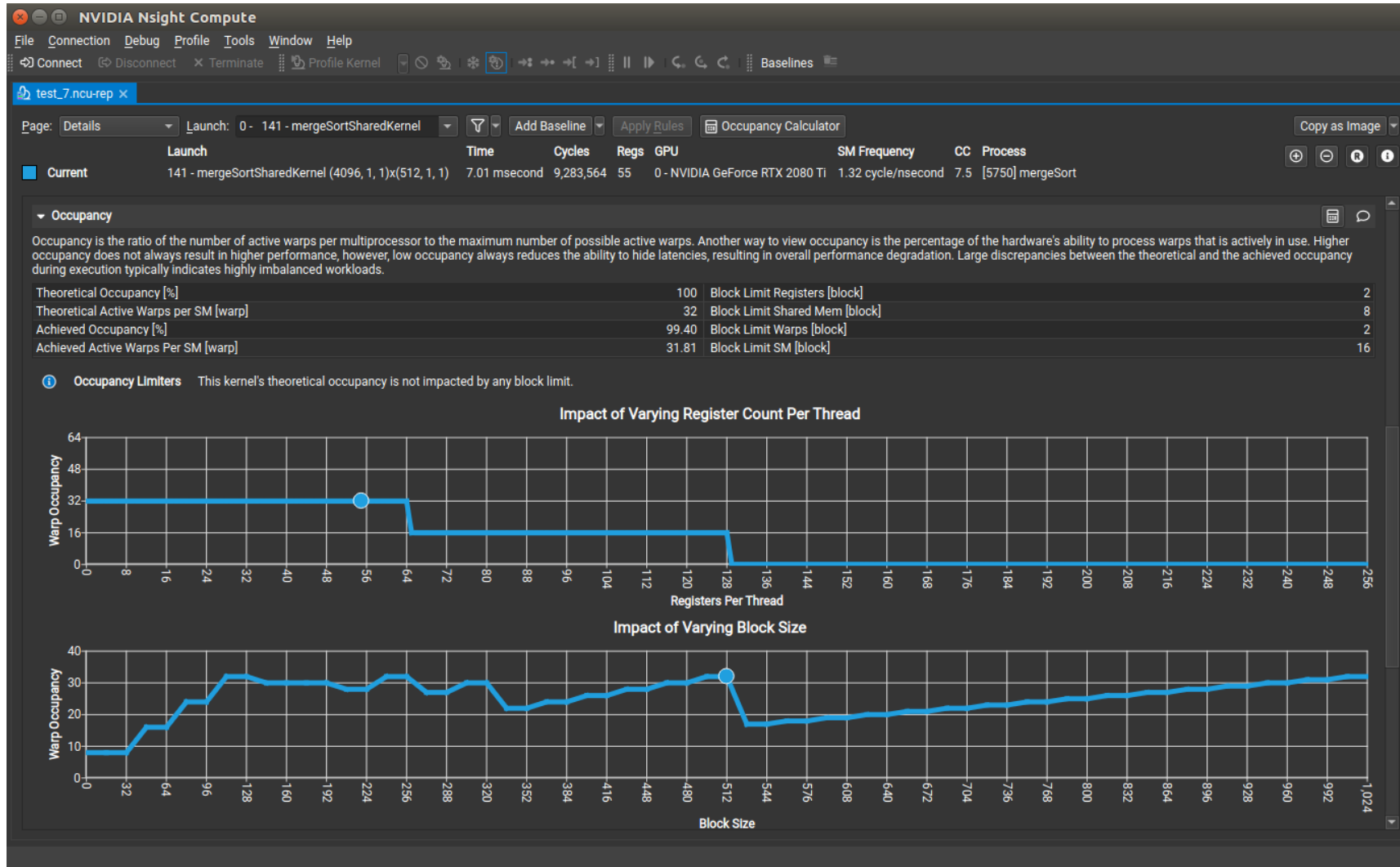
#	Source	Sampling Data (All)	Instructions Executed
234	<code>typename DType, typename OType></code>	0	6,144
235	<code>__global__ void softmax_compute_kernel(DType *in, OType *out, index_t M</code>	0	6,144
236	<code>Shape<ndim> sshape, Shape<ndim></code>	0	6,144
237	<code>const double temperature) {</code>	0	6,144
238	<code>const unsigned x_size = 1 << x_bits;</code>	0	6,144
239	<code>__shared__ AType smem[x_size];</code>	0	6,144
240	<code>index_t sa = stride[axis];</code>	61	6,144
241	<code>index_t base = unravel_dot(blockIdx.x, sshape, stride);</code>	0	6,144
242	<code>index_t x = threadIdx.x;</code>	52	6,144
243	<code></code>	0	6,144
244	<code>red::maximum::SetInitValue(smem[x]);</code>	44	6,144
245	<code>for (index_t i = x; i < M; i += x_size) {</code>	9	6,144
246	<code>smem[x] = ::max(smem[x], negate ? -in[base + i*sa] : in[base + i*sa</code>	6	6,144
247	<code>};</code>	0	6,144
248	<code>__syncthreads();</code>	7	6,144
249	<code>cuda::Reduce1D<red::maximum, x_bits>(smem);</code>	111	6,144
250	<code>};</code>	0	6,144
251	<code>void softmax_compute_kernel(DType *in, OType *out, index_t M</code>	0	6,144
252	<code>Shape<ndim> sshape, Shape<ndim></code>	0	6,144
253	<code>const double temperature) {</code>	0	6,144
254	<code>const unsigned x_size = 1 << x_bits;</code>	0	6,144
255	<code>DType val;</code>	14	6,144
256	<code>for (index_t i = x; i < M; i += x_size) {</code>	118	6,144
257	<code>val = negate ? -in[base + i*sa]:in[base + i*sa];</code>	0	6,144
258	<code>smem[x] += static_cast<AType>(expf((val - smax) / static_cast<AType</code>	0	6,144
259	<code>};</code>	0	6,144
260	<code>__syncthreads();</code>	0	6,144
261	<code>cuda::Reduce1D<red::sum, x_bits>(smem);</code>	208	6,144
262	<code>__syncthreads();</code>	0	6,144
263	<code>AType ssum = smem[x];</code>	0	6,144
264	<code>__syncthreads();</code>	1	6,144
265	<code></code>	0	6,144
266	<code>for (index_t i =</code>	7	6,144
267	<code>val = negate ? -in[base + i*sa] : in[base + i*sa];</code>	14	6,144
268	<code>out[base + i*sa] = OP::Map((val - smax)/static_cast<DType>(temperat</code>	0	6,144
269	<code>};</code>	0	6,144

PTX/SASS Summary:

#	Source	Sampling Data (All)	Instructions Executed
133	<code>BSYNC B0</code>	0	6,144
134	<code>NOP</code>	0	6,144
135	<code>BAR.SYNC 0x0</code>	1	6,144
136	<code>ISETP.GT.AND P0, PT, R11, 0x3f, PT</code>	2	6,144
137	<code>BSSY B1, 0x7f87d326fc50</code>	1	6,144
138	<code>ISETP.GT.AND P1, PT, R11, 0x1f, PT</code>	1	6,144
139	<code>ISETP.GT.AND P2, PT, R11, 0xf, PT</code>	0	6,144
140	<code>ISETP.GT.AND P3, PT, R11, 0x7, PT</code>	2	6,144
141	<code>@!P0 LDS.U R4, [R14+0x100]</code>	2	6,144
142	<code>@!P0 LDS.U R5, [R14]</code>	4	6,144
143	<code>@!P0 STL [R1+0x8], R4</code>	3	6,144
144	<code>@!P0 FMMX R5, R5, R4, !PT</code>	2	6,144
145	<code>@!P0 STS [R14], R5</code>	4	6,144
146	<code>NOP</code>	0	6,144
147	<code>BAR.SYNC 0x0</code>	4	6,144
148	<code>@!P1 LDS.U R6, [R14+0x80]</code>	8	6,144
149	<code>ISETP.GT.AND P0, PT, R11, 0x3, PT</code>	0	6,144
150	<code>P1 LDS.U R7, [R14]</code>	1	6,144
151	<code>P1 STL [R1+0xc], R6</code>	4	6,144
152	<code>P1 FMMX R7, R7, R6, !PT</code>	0	6,144
153	<code>P1 STS [R14], R7</code>	3	6,144
154	<code>NOP</code>	4	6,144
155	<code>BAR.SYNC 0x0</code>	0	6,144
156	<code>@!P2 LDS.U R8, [R14+0x40]</code>	9	6,144
157	<code>ISETP.GT.AND P1, PT, R11, 0x1, PT</code>	0	6,144
158	<code>@!P2 LDS.U R9, [R14]</code>	2	6,144
159	<code>@!P2 STL [R1+0x10], R8</code>	0	6,144
160	<code>@!P2 FMMX R9, R9, R8, !PT</code>	0	6,144
161	<code>@!P2 STS [R14], R9</code>	0	6,144
162	<code>NOP</code>	0	6,144
163	<code>@!P3 LDS.U R10, [R14+0x20]</code>	4	6,144
164	<code>@!P3 LDS.U R5, [R14]</code>	0	6,144
165	<code>@!P3 STL [R1+0x14], R10</code>	4	6,144
166	<code>@!P3 FMMX R5, R5, R10, !PT</code>	2	6,144
167	<code>@!P3 STS [R14], R5</code>	2	6,144
168	<code>NOP</code>	0	6,144

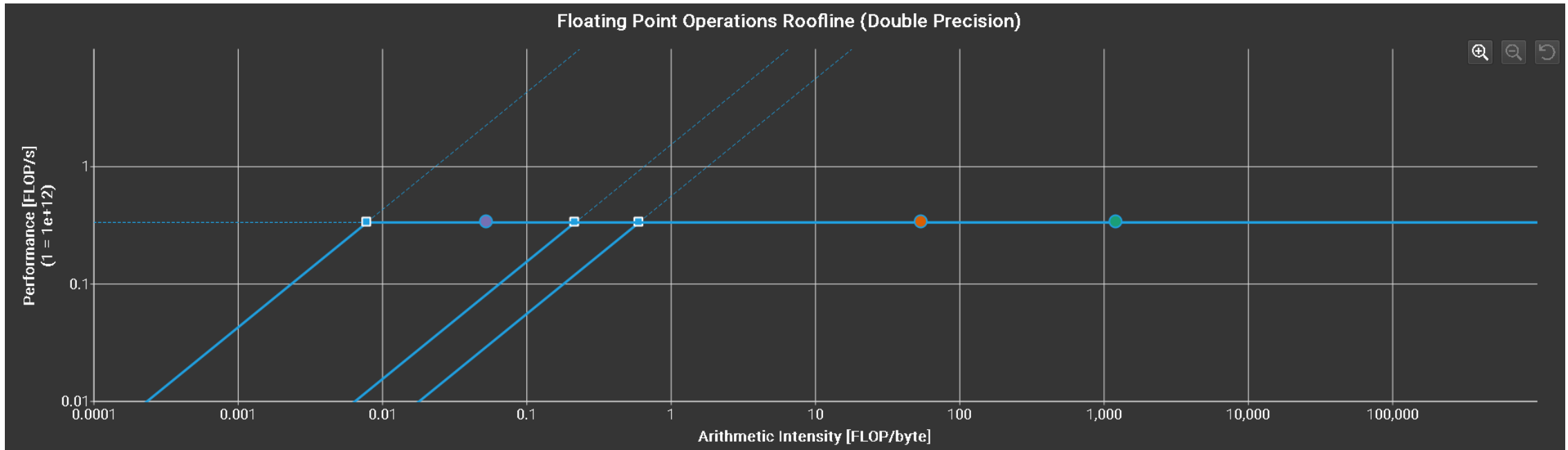
OCCUPANCY CALCULATOR

Model Hardware Usage and Identify Limiters



- Model theoretical hardware usage
- Understand limitations from hardware vs. kernel parameters
- Configure model to vary HW and kernel parameters
- Opened from an existing report or as a new activity

HIERARCHICAL ROOFLINE



- Visualize multiple levels of the memory hierarchy
- Identify bottlenecks caused by memory limitations
- Determine how modifying algorithms may (or may not) impact performance

Sections/Rules Info			
Sections/Rules			
Enter filter			
	Name	Priority	Description
<input checked="" type="checkbox"/>	GPU Speed Of Light Throughput (1)	10	High-level overview of the throughput for compu...
<input checked="" type="checkbox"/>	GPU Speed Of Light Roofline Chart (1)	11	High-level overview of the utilization for comput...
<input checked="" type="checkbox"/>	GPU Speed Of Light Hierarchical Roofline Chart (Double Precision)	12	High-level overview of the utilization for comp...
<input checked="" type="checkbox"/>	GPU Speed Of Light Hierarchical Roofline Chart (Half Precision)	12	High-level overview of the utilization for comput...
<input checked="" type="checkbox"/>	GPU Speed Of Light Hierarchical Roofline Chart (Single Precision)	12	High-level overview of the utilization for comput...
<input checked="" type="checkbox"/>	GPU Speed Of Light Hierarchical Roofline Chart (Tensor Core)	12	High-level overview of the utilization for comput...
<input type="checkbox"/>	Compute Workload Analysis (2)	20	Detailed analysis of the compute resources of t...

KERNEL PROFILES WITH NSIGHT COMPUTE

```
$ ncu -k mykernel -o report ./myapp.exe
```

- Generated file: report.ncu-rep
 - Open for viewing in the Nsight Compute UI
- (Without the -k option, Nsight Compute will profile everything and take a long time)

CUDA-GDB

Command-Line and IDE Back-End Debugger

- Unified CPU and CUDA Debugging
- CUDA-C/SASS support
- Built on GDB and uses many of the same CLI commands
- Local/Remote connection support

```
(cuda-gdb) info cuda threads breakpoint all
  BlockIdx ThreadIdx          Virtual PC Dev SM Wp Ln      Filename  Line
Kernel 0
  (1,0,0)   (0,0,0) 0x0000000000948e58   0 11  0  0 infoCommands.cu  12
  (1,0,0)   (1,0,0) 0x0000000000948e58   0 11  0  1 infoCommands.cu  12
  (1,0,0)   (2,0,0) 0x0000000000948e58   0 11  0  2 infoCommands.cu  12
  (1,0,0)   (3,0,0) 0x0000000000948e58   0 11  0  3 infoCommands.cu  12
  (1,0,0)   (4,0,0) 0x0000000000948e58   0 11  0  4 infoCommands.cu  12
  (1,0,0)   (5,0,0) 0x0000000000948e58   0 11  0  5 infoCommands.cu  12

(cuda-gdb) info cuda threads breakpoint 2 lane 1
  BlockIdx ThreadIdx          Virtual PC Dev SM Wp Ln      Filename  Line
Kernel 0
  (1,0,0)   (1,0,0) 0x0000000000948e58   0 11  0  1 infoCommands.cu  12
```

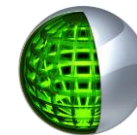
COMPUTE SANITIZER

Automatically Scan for Bugs and Memory Issues

- Compute Sanitizer checks correctness issues via sub-tools:
 - **Memcheck** - Memory access error and leak detection tool.
 - **Racecheck** - Shared memory data access hazard detection tool.
 - **Initcheck** - Uninitialized device global memory access detection tool.
 - **Synccheck** - Thread synchronization hazard detection tool.
- <https://github.com/NVIDIA/compute-sanitizer-samples>

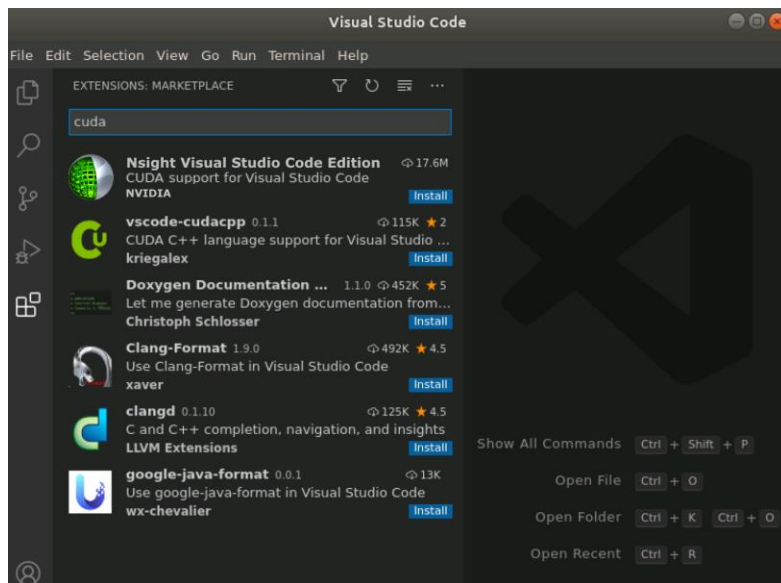
```
$ make run_memcheck
/usr/local/cuda/compute-sanitizer/compute-sanitizer --destroy-on-device-error kernel memcheck_demo
===== COMPUTE-SANITIZER
Mallocing memory
===== Invalid __global__ write of size 4 bytes
=====   at 0x70 in unaligned_kernel()
=====   by thread (0,0,0) in block (0,0,0)
=====   Address 0x7f671ac00001 is misaligned
=====   and is inside the nearest allocation at 0x7fb654c00000 of size 4 bytes
=====   Saved host backtrace up to driver entry point at kernel launch time
=====   Host Frame: [0x2774ec]
=====         in /lib/x86_64-linux-gnu/libcuda.so.1
=====   Host Frame: __cudart803 [0xfccb]
=====         in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: cudaLaunchKernel [0x6a578]
=====         in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: cudaError cudaLaunchKernel<char>(char const*, dim3, dim3, void**, unsigned
=====         in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: __device_stub_Z16unaligned_kernelv() [0xb22e]
=====         in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: unaligned_kernel() [0xb28c]
=====         in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: run_unaligned() [0xaf55]
=====         in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: main [0xb0e2]
=====         in /home/cuda/github/compute-sanitizer-samples/Memcheck/memcheck_demo
=====   Host Frame: ../sysdeps/nptl/libc_start_call_main.h:58: __libc_start_call_main [0x2dfd0]
=====         in /lib/x86_64-linux-gnu/libc.so.6
```

NSIGHT VISUAL STUDIO CODE EDITION



Visual Studio Code extensions that provides:

- CUDA code syntax highlighting
- CUDA code completion
- Build warning/errors
- Debug CPU & GPU code
- Remote connection support via SSH
- Available on the VS Code Marketplace now!



Visual Studio Code interface with the CUDA debugger active. The interface is annotated with callouts:

- Variables view:** Shows local variables such as Bs, As, C, A, wA, wB, a, b, tx, aStep, bx, ty, aBegin, aEnd, bBegin, bStep, and Csub.
- CPU & GPU registers:** Shows registers R0, R1, and R2.
- Watch CPU & GPU vars:** Shows the variable 'As' with a value of [32].
- Session status:** Shows the status of the debug session.
- Exec debugger commands:** Shows the command '-exec info cuda warps' in the debug console.
- CUDA Call Stack:** Shows the call stack for the current thread, including 'MatrixMulCUDA-32', 'matrixMul', 'matrixMul', 'cuda-EvtHandlr', and 'matrixMul'.
- CUDA focus:** Shows the current thread and warp information in the status bar.

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

ADDITIONAL RESOURCES

- Sessions
 - [A41100](#) - CUDA: New Features and Beyond
 - [A41131](#) - Developing Efficient CUDA Kernels for Fourth-Generation Tensor Cores
- Labs
 - [DLIT41277](#) - Optimizing CUDA Machine Learning Codes with Nsight Profiling Tools
 - [DLIT41274](#) - Debugging and Analyzing Correctness of CUDA Applications
 - [DLIT41276](#) - Developer Tools Fundamentals for Ray Tracing using NVIDIA Nsight Graphics and NVIDIA Nsight Systems
- Ampere Architecture Detailed Blog
 - [NVIDIA Ampere Architecture In-Depth](#)
- Developer Tools are free and packaged in the latest version of the CUDA Toolkit
 - <https://developer.nvidia.com/cuda-downloads>
- Support is available via:
 - <https://forums.developer.nvidia.com/c/development-tools/>
- More information at:
 - <https://developer.nvidia.com/tools-overview>

HANDS-ON


/lus/eagle/projects/SDL_Workshop/jacobi

- Solving Laplace Equation with Jacobi Iterations

```

$$f_{i,j} = (f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}) / 4$$
  
  
// set initial conditions at f[0][:], f[N-1][:], f[:,0], f[:,N-1]  
  
while (error > tolerance):  
    error = 0  
    for i = 1, N-2:  
        for j = 1, N-2:  
            f[i][j] = 0.5 * (f_old[i+1][j] + f_old[i-1][j] + f_old[i][j+1] + f_old[i][j-1])  
            error += (f[i][j] - f_old[i][j]) * (f[i][j] - f_old[i][j])  
    swap(f_old, f)
```

WWW.OPENHACKATHONS.ORG

 Home Events ▾ Attendees ▾ Mentors ▾ About ▾ OpenACC

All HACKATHONS BOOTCAMPS All Locations ▾

KMA and NOAA Workshop on AI for Weather and Climate

Date(s): Oct 21, 2022 - Oct 21, 2022
Event Focus: **HPC+AI**
Asia-Pacific



Applications [Open](#)

NVIDIA/HLRS SciML GPU Bootcamp

Date(s): Oct 24, 2022 - Oct 25, 2022
Event Focus: **HPC+AI**
Europe/Middle East/Africa



Applications [Open](#)

NCI GPU Hackathon 2022

Date(s): Oct 25, 2022 - Nov 4, 2022
Event Focus: **HPC+AI**
Asia-Pacific



Applications **Closed**

TWCC Open Hackathon 2022

Date(s): Nov 23, 2022 - Dec 2, 2022
Event Focus: **HPC+AI**
Asia-Pacific



Applications [Open](#)

NERSC GPU Hackathon 2022

Date(s): Nov 30, 2022 - Dec 8, 2022
Event Focus: **HPC+AI**
North America/Latin America



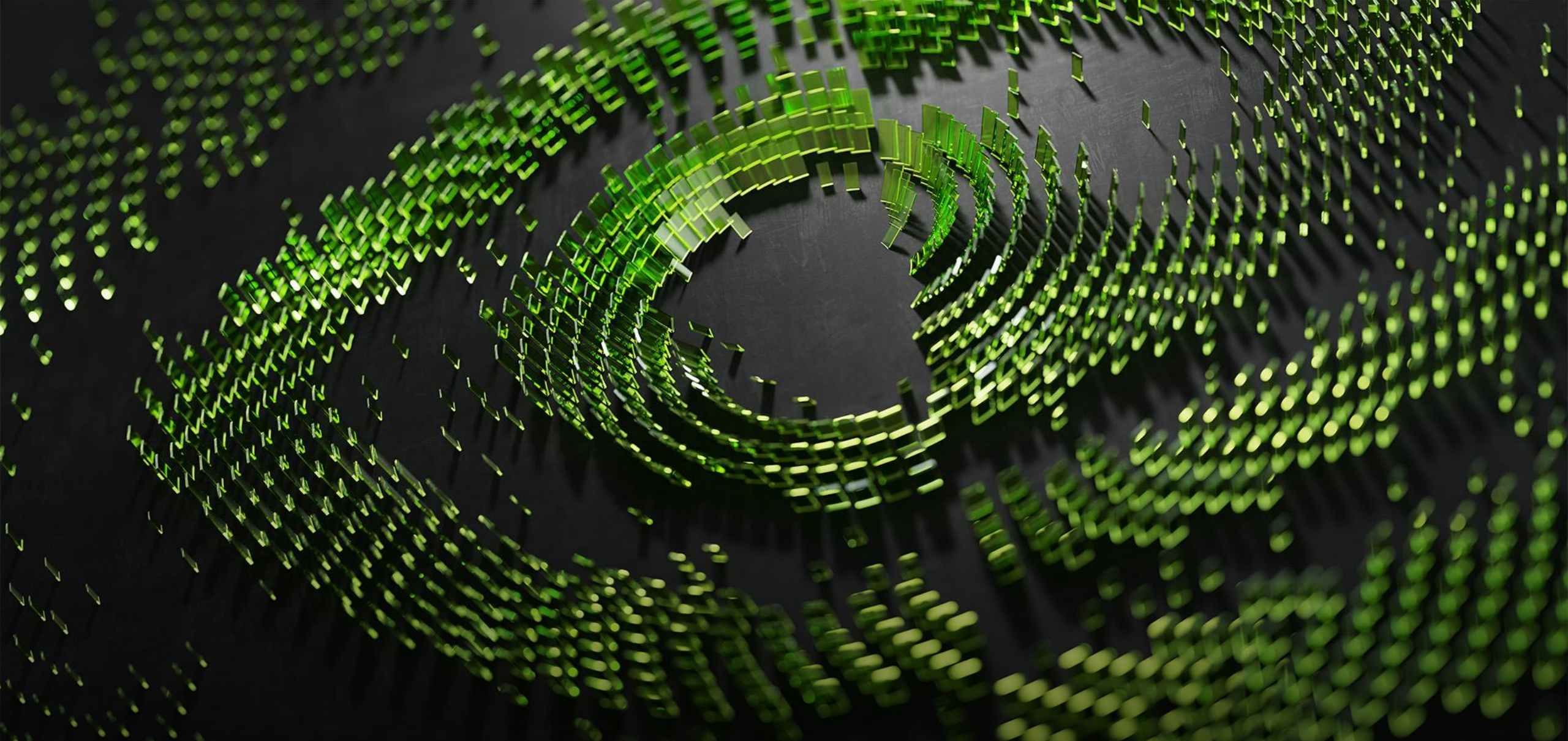
Applications [Open](#)

Westlake University GPU Hackathon 2022

Date(s): Nov 29, 2022 - Dec 9, 2022
Event Focus: **HPC+AI**
Asia-Pacific



Applications [Open](#)



nVIDIA®