

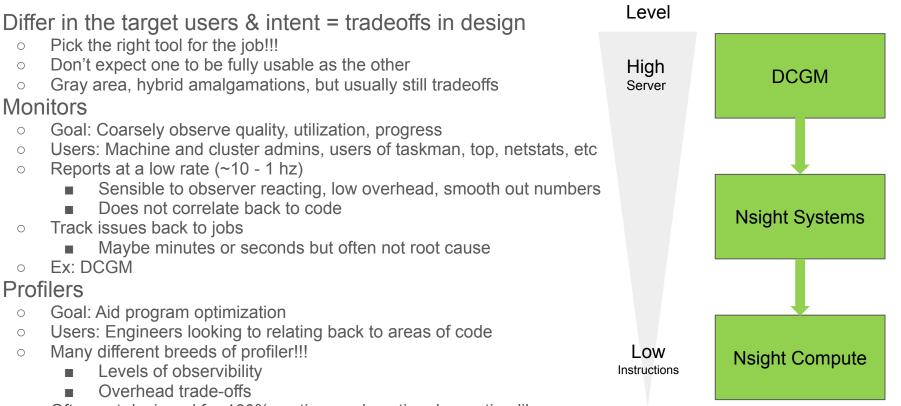
Profiling Deep Learning with Nsight Systems

Daniel Horowitz Director of Engineering Platform Developer Tools

Agenda

- Monitors vs Profilers
- Profiling on a single node
 - Overview
 - Basic features
 - GPU investigations
 - CPU investigations & features
 - Common bubble recovery tactics
- Profiling multi-node

Monitors vs Profilers



0 Monitors

0

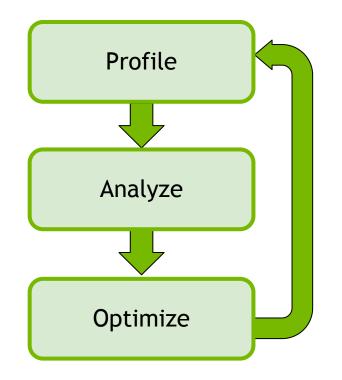
Ο

- Goal: Coarsely observe quality, utilization, progress 0
- Users: Machine and cluster admins, users of taskman, top, netstats, etc 0
- Reports at a low rate (~10 1 hz) Ο

Pick the right tool for the job!!!

- Sensible to observer reacting, low overhead, smooth out numbers
- Does not correlate back to code
- Track issues back to jobs 0
 - Maybe minutes or seconds but often not root cause
- Fx[·] DCGM 0
- Profilers
 - Goal: Aid program optimization Ο
 - Users: Engineers looking to relating back to areas of code Ο
 - Many different breeds of profiler!!! Ο
 - Levels of observibility
 - Overhead trade-offs
 - Often not designed for 100% up-time and continual reporting like a Ο monitor
 - Ex: Nsight Systems and Nsight Compute Ο

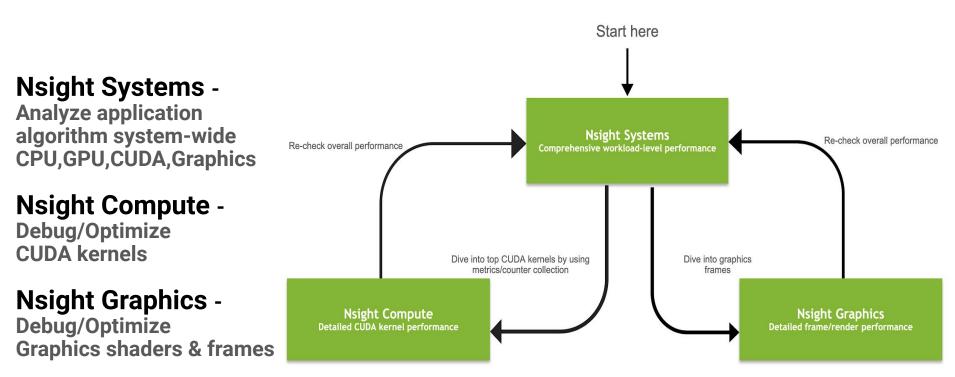
Profiler-Driven Optimization Workflow



Iterate until desired performance is achieved

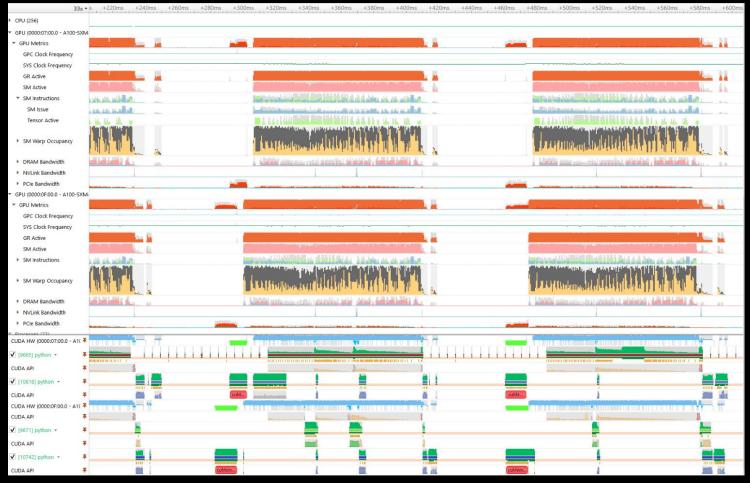
Nsight Profiler Family

Performance Analysis Workflow

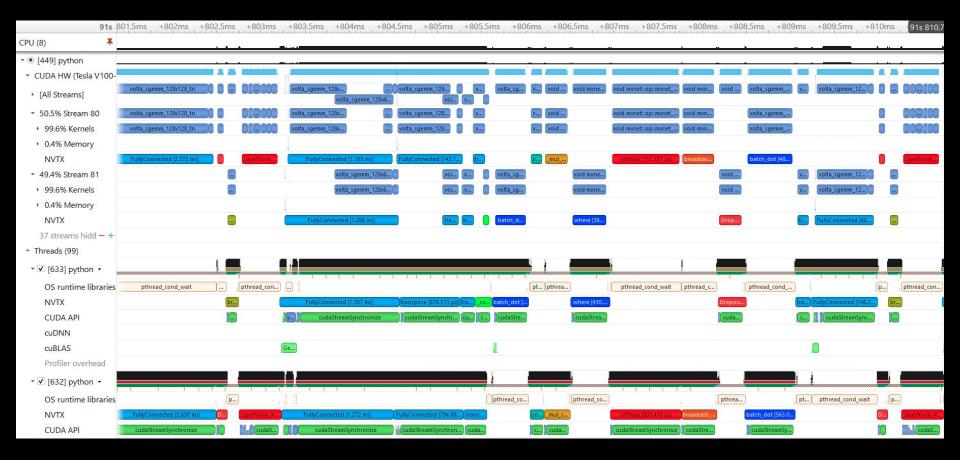


Nsight Systems





Timeline looks like a spectrogram when zoomed out Ex: DL model training, 2 iterations, GPU & CPU



Zooming in transforms many row graphs into ranges



Tuning an Orchestra of Tasks













Why start with Nsight Systems?

It's designed to allow you to...

See the big picture

See how asynchronous CPUs, GPUs, NICs, and software are interacting Who is stuck on whom

Measure the higher-level costs to pick the best opportunities

Avoid work based on intuition & false-positive indicators

Don't assume GPU bound and skip ahead to other tools

Some synchronization oriented issues can look like GPU bound

Statistics alone often aren't enough info to understand and resolve the issue

WARNING: The presenter & slides will often refer to "Nsight Systems" as Nsys or NSYS

Overview

- System-wide application algorithm tuning
- Locate optimization opportunities
 - Visualize millions of events on a very fast GUI timeline
 - See gaps of unused CPU and GPU time
- Balance your workload across multiple GPUs, CPUs, NICs
 - GPU streams, kernels, memory transfers, etc
 - CPU algorithms, utilization, and thread state
- UX: CLI, GUI timeline, statistics, data export
 - Ex: Collect nsys-rep w/CLI on cluster, SCP to PC to view, mine either place
- Multi-platform: Linux & Windows, x86-64, ARM server, Tegra
 - Mac (host-only)

Timeline Features

GPU

GPU HW metrics sampling

CUDA GPU-side kernel and mem-op ranges correlated to CPU API calls

Libraries: cuBLAS, cuDNN, cuDF, TensorRT

CPU, OS, & application

Thread state, migrations, and call-stacks

OS runtime long call trace (>1us, pthread, glibc \rightarrow mmap, file & IO, ...)

Call-stack backtraces (>80us)

ftrace or WDDM & ETW (page faults, signal, interrupts, ...)

Code Annotations APIs

NVTX

Networking

UCX, MPI (OpenMPI & MPICH), OpenSHMEM, NVSHMEM, NCCL trace NVIDIA NIC/HCA metrics sampling (Infiniband & Ethernet)

Wait!!! I just want stats about my DNN 😕

You have barely mentioned Deep Learning!!!

Each DL framework has its own tools, so we'll discuss how to go deeper Many DL frameworks have their DNN layer execution instrumented with NVTX

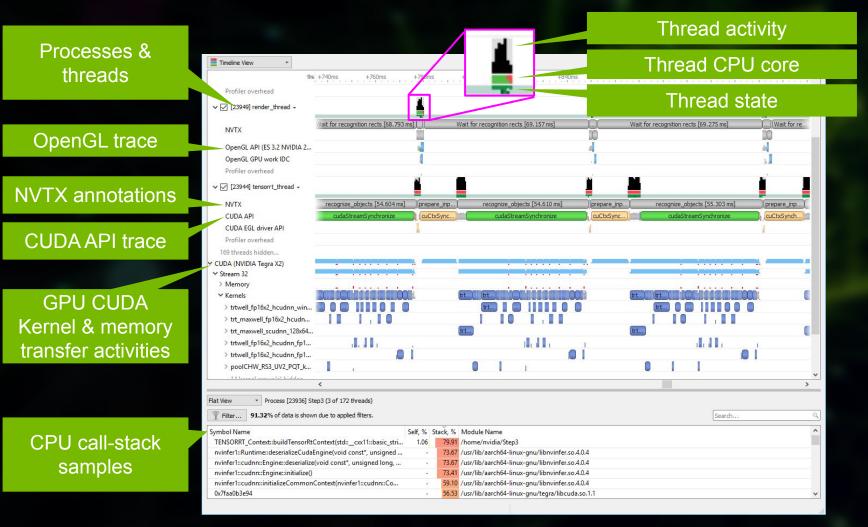
Sometimes just the NVIDIA container if the framework doesn't accept it upstream Nsys has the relationships: DNN layer \rightarrow CUDA Launch \rightarrow GPU CUDA Kernel Execution Nsys can export to SQLite

Nsys has python scripts & documentation on how to analyze that database

If that's all you want, that's the easy path

But we're about to go deeper and show you how it's executing!!!

So you can visually investigate, craft better stats, and create your own expert systems

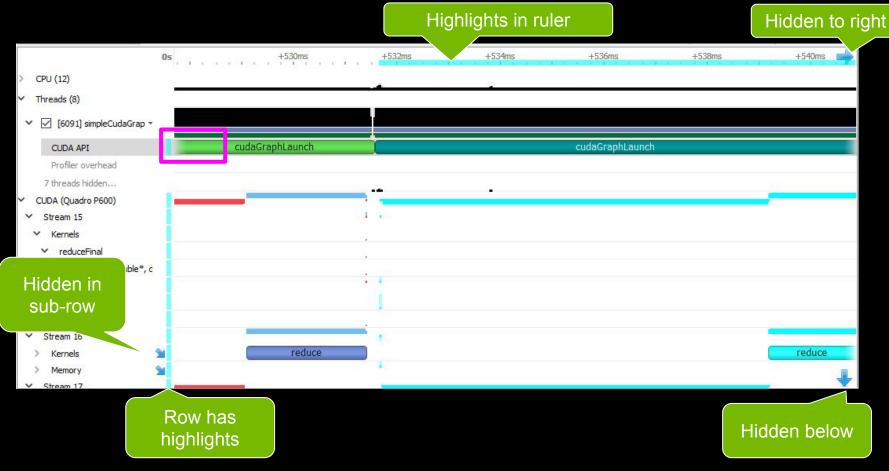


CUDA

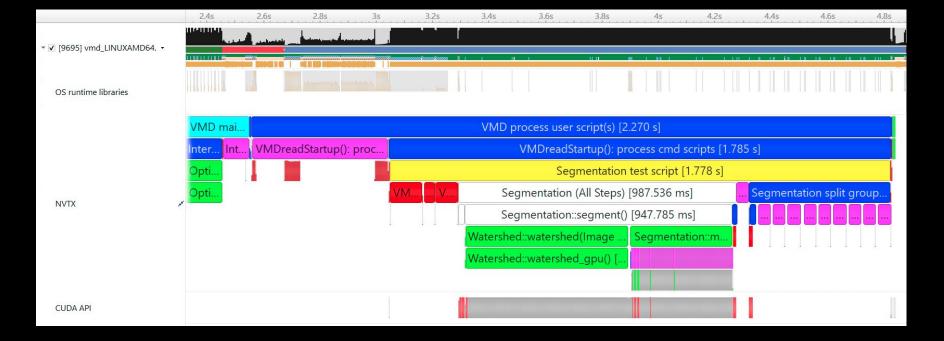
	3s +209.8ms	+209.85ms	+209.9ms	+209.9	5ms +21	Oms +210,05	ms +210.1ms	+210.15ms	+210.2ms	+210.25ms	+210.3ms	+210.35ms	+210.4ms	+210. 3s 21
CPU (80)													14291, 2017 - 272 - 2016 - 2026	
[413] python														
OS runtime libraries			pthre	_	_									
CUDA API	cuMemcpy	HtoDAsync		cuMemcp	split_v_kernel									
✓ CUDA (Tesla P100-SXM2-16GB, 0				_ L										
✓ Stream 70														
> Memory														
✓ Kernels									split_v_ke	ernel				

API launch to GPU kernel correlation

NVIDIA



CPU-GPU correlation & location assistance

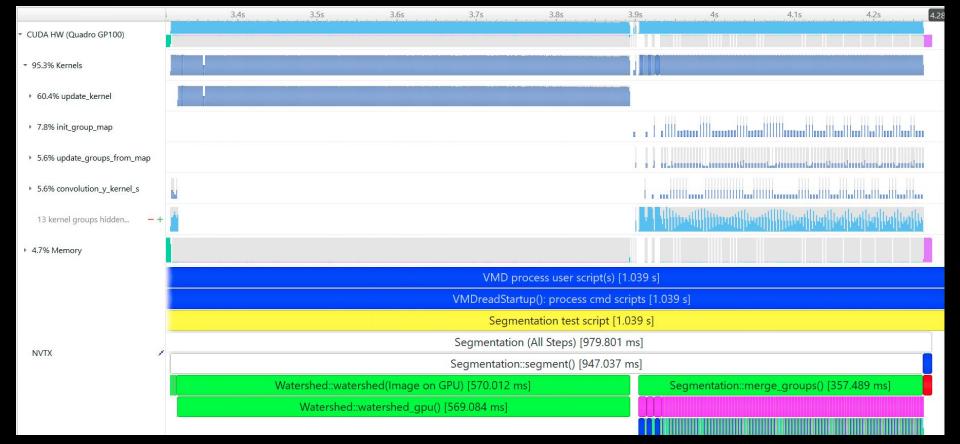


Code annotations APIs NVTX = <u>NV</u>IDIA <u>T</u>ools e<u>X</u>tensions

Example: Visual Molecular Dynamics (VMD) algorithms visualized with <u>NVTX</u> on CPU

7

NVIDIA



NVTX ranges project from CPU onto the GPU CUDA streams

Example: Visual Molecular Dynamics (VMD) algorithms visualized with NVTX on GPU

PyTorch

DNN layer annotations are disabled by default

Add the following: "with torch.autograd.profiler.emit_nvtx():"

TensorRT is also annotated already if that is the backend you are using

You can also add NVTX to your python script manually:

https://github.com/pytorch/pytorch/blob/master/torch/cuda/nvtx.py https://pytorch.org/docs/stable/generated/torch.cuda.nvtx.range_push.html https://pytorch.org/docs/stable/generated/torch.cuda.nvtx.range_pop.html

TensorFlow

DNN layers are annotated <u>by default</u> with NVTX in NVIDIA TF containers! <u>https://catalog.ngc.nvidia.com/orgs/nvidia/containers/tensorflow</u> <u>https://github.com/NVIDIA/tensorflow</u> TF_DISABLE_NVTX_RANGES=1 if you want to disable for production

Adding more detail to the timeline for setup,eager mode, tf.data.Dataset.from_generator, etc <u>https://github.com/NVIDIA/NVTX/tree/dev/python</u> nsys profile --trace=cuda,nvtx,osrt,cudnn,cublas --backtrace=dwarf --capture-range=cudaProfilerApi --gpu-metrics-devices=all -output=oft-profile-dwarf4 sh scripts/expt-singleGPU.sh --profile 50 --profile start 5000 --profile epoch 1

Nsight Systems CLI command

Select APIs to trace

Enable GPU memory use tracking (but there is extra overhead)

Collect thread call-stack sample backtraces via DWARF info - deeper but more expensive to collect Trigger collection on cudaProfilerStart API in application, or consider timer-based options GPU metrics sampling at default 10khz

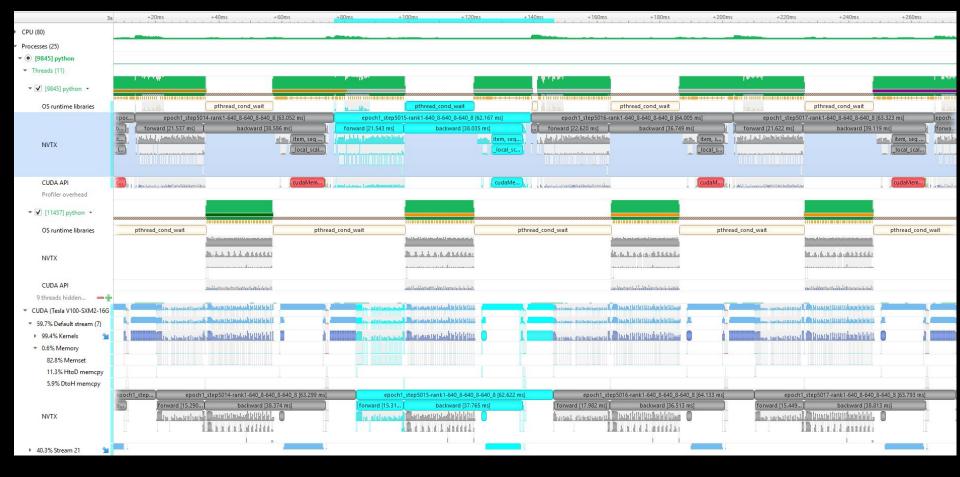
Name the report file

Application command - plus arguments for when to start profiling

https://docs.nvidia.com/nsight-systems/UserGuide/index.html#cli-profile-command-switch-options

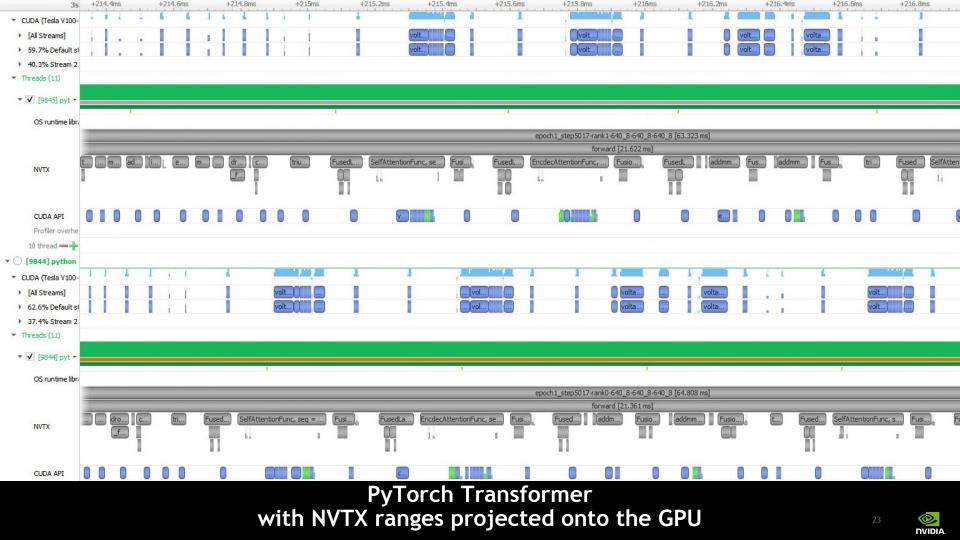
srun nsys profile ... required on multi-node or multi-container nsys profile mpirun ... optional on single node to produce a single report

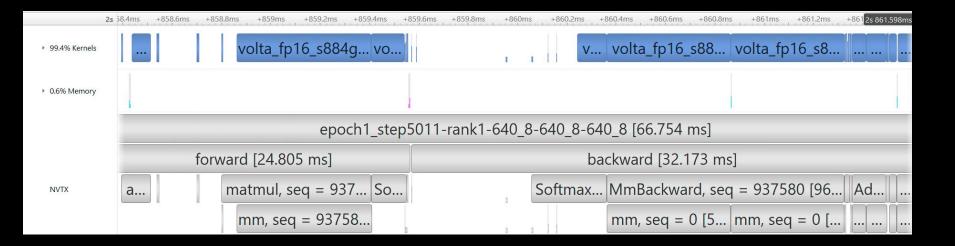
Nsight Systems CLI command line example



PyTorch Transformer with NVTX ranges projected onto the GPU

NVIDIA





Example: NVTX on GPU in PyTorch transformer model in eager mode (ie non-hybrid)

24

NVIDIA

235	s +653,4ms +653,6ms +653,8ms +654,6ms +654,2ms +654,4ms +654,6ms +654,8ms +655,5ms +655,7ms +655,4ms +655,6ms +655,8ms +655,8ms +655,2ms
CUDA (Tesla V100-SXM2-16GB, I	
✓ Stream 126	
> Kernels	[nh] (EgenMetakemel)(ReluGradHalKemel), b_ bw_1C11_sngleread_p16](ndrwToNh)(volta_fp16_s884cudm(ndrwToNh)(volta_s884cudm)(ndrwToNh)(volta_s884cudm)(ndrwToNh)(volta_fp16_s884cudm)(volta_fp16_s884cudm)(ndrwToNh)(volta_fp16_s884cudm)(volta_fp16_
> Memory	
V NVTX	
Async Ranges	Conv _ AddN: GPU_0/to_ Reluciad: GPU_0_ FusedBatchNomGradV2: GP [Conv2DBadgropFile: GPU_0/tower_] Conv2DBadgropFile: GPU_0/tower_0/g_ R Fused [Conv2DBadgropFile: GPU_0/tower_0/g_ R] Fused] Conv2DBadgropFile: GPU_0/tower_0/g_ R] Fused] Fused] Fused
> Stream 129	
16 streams hidden	
CUDA (Tesla V100-SXM2-16GB, I	
✓ Stream 130	
> Kernels	dgrad_engine scalePackedTensor_kernel dgrad_engine
> Memory	
V NVTX	
Async Ranges	Conv2DBadgropInput: GPU_1/kower_1/gradients/GPU_1/kower_1/resnet_v1_7/conv25/Conv2DBadgropInput [9.034 ms]
15 streams hidden	
CUDA (Tesla V100-SXM2-16GB, I	
✓ Stream 134	
> Kernels	m. (R) [bn.b] wolta_fp16_s884cudm_fp16_256x128_]n. wolta_s884cudm_fp16_128x128 (R) [bn.b] (
> Memory	
V NVTX	
Async Ranges	com. R. Fused Conv2DBadgrop.Fiter: GPU_2/tower_2/grade Conv2DBadgrop.Fiter: GPU_2/tower_2/grade Conv2DBadgrop.Fiter: GPU_2/to Conv2DBadgrop.Fiter: GPU_2/tower_2/grade FusedBatdNormGradV2: GP Conv2DBadgrop.Fiter: GPU_2/tower_2/grade
16 streams hidden	
CUDA (Tesla V100-SXM2-16GB,	
✓ Stream 138	
> Kernels	volta_fp16_scudm_fp16_128x128_stridedB_splitK_small_m_y1
> Memory	
V NVTX	
Async Ranges	Conv2D8adgropFilter: GPU_3/tower_3/gradents/GPU_3/tower_3/resnet_v1_13/conv45/Conv2D8adgropFilter (30.542 ms)

TensorFlow Resnet50 DNN nodes as NVTX ranges projected onto the GPU



■ Timeline View					Q 1x □	i i i	① <u>23 messages</u>	
15	+521,52ms +521,54ms +521,56ms +521,5	58ms +5211s 521,6	5087ms 1,62ms +521,64ms +521,66ms	+521,68ms	+521,7ms	+521,72ms	+521,74ms	
▼ [19621] NVSHMEM PE 0	1063 px; 0 msec							
NCCL	1063 px; 0 msec	ncclAllReduce [26,3)					
NVSHMEM	[nvshmem_malloc [36,690 μs]] 1063 px; 0 m[n]	nvshmemx_int_sum_r.			nvs	nvshmem_free hmem_quiet []	[91,979 µs]	
CUDA API	(c cuSt) ba cuStre accu	ncclKe]	cudaStreamSynchronize	cud	00	cuStreamSy	arr) cuStrea	
Profiler overhead	1063 px; 0 msec							
▼ CUDA HW (0000:06:00.0 -	1 1063 px msec							
 [All Streams] 	1063 px msec bar a	0	cclKernel_AllReduce_RING_LL_S				barrie	
79.5% Default stream 7	1063 px; 0 msec		ncclKernel AllReduce RING LL Sum int32 t					
7.335 ⁺ 19.8% Stream 22	1063 px and msec bar		Begins: 1,52161s Ends: 1,52165s (+41,504 μs)				barrie	
Events View 👻			grid: <<<1, 1, 1>>> block: <<<64, 1, 1>>> Launch Type: Regular Static Shared Memory: 41.472 bytes Dynamic Shared Memory: 0 bytes	[Name -	•	٩,	
# • Name			Registers Per Thread: 96	TID	 Descr 	ription:		
🛾 730 🔹 🗌 nvshm	emx_int_sum_reduce_on_stream		Local Memory Per Thread: 0 bytes Local Memory Total: 251.658.240 bytes	19621	nvshmemx_int_sum_redu		educe_on_stream	
🛄 731 🔹 🗍 ncc	IAIIReduce		Shared Memory executed: 98.304 bytes	19621		Begins: 1,52158s Ends: 1,52161s (+28,855 μs)		
732 CL	daStreamIsCapturing		Shared Memory Bank Size: 4 B Launched from thread: 19621		Thread: 19621			
733 CL	daStreamWaitEvent		Latency: ←10,563 µs	19621				
734 no	clKernel_AllReduce_RING_LL_Sum_int32_t		Correlation ID: 4189 Stream: Stream 20	19621	-			

NVSHMEM & NCCL

▼ [1730455] p	ython							
DALI]				[DAL	[DALI][Executor] RunGPU [716.997 µs] I][GP[[DALI][GPU op]Bo[][DALI]]		
CUDA API					1			
9 threads hi	dden	-+					•	
CUDA HW (A1	00-SXM-80GB - 0	000:4;		1	💼 🗮 11		t.	T 1
[All Streams]	1	1	ncdKernel_AllReduce_RING_LL_Sum_ha	f bat		Bat B Ba B Li	Encode	M SliceFlipN
▶ 32.1% Stree	am 109							
▶ 30.5% Stre	am 33	1						0 6
▶ 13.2% Stree	am 94		ncdKernel_AllReduce_RING_LL_Sum_ha	f bat nc				
▶ 10.2% Stree	am 110							
1.8% Stream	n 108							
▶ 1.4% Stream	n 107							
* 1.2% Stream	m 29					Bat B) Ba B) Li	Encode	M SliceFlipN
▶ 89.4% Ke	rnels					Bat B B Li	Encode	M SliceFlipN
▶ 10.6% Me	mory							1 B
DALL						[DAI	.I][Executor] RunGPU [1.639 ms]	
DALI						[DALI][GPU op]Re] [D] [DA	LI][GPU op]BoxEncoder_13 [1.031 ms]	[D] [DALI] [GP]
▶ 0.8% Stream	n 106							

NVIDIA DALI trace Data Loading Library



NVIDIA.

Core Investigation Strategy

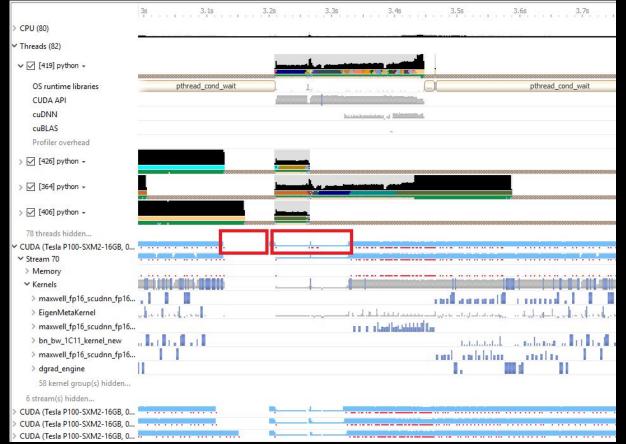
- What's HOT?
 - Will it be easier to shrink what i have?
 - This is where MOST people concentrate. ...intuitive but not always better

What's COLD?

- Will it be easier to take advantage of the something unused?
- Free money? Yes please!
- Is it doing what I intended & budgets? (hint: often not as well as you thought)
- Cold spots are often clear, measurable opportunities!!!
 - How can i remove or fill them?
 - Where do i have incorrect/unnecessary/unexpected dependencies/synchronization?
- Hot spots
 - Might be parallelizable?
 - Might not be shrinkable without compromising accuracy, memory, etc

GPU Bubble Detective

- Find the crime (cold or cool spot)
- Use correlation to track back to the CPU
 - Select surrounding GPU CUDA ops
- Investigate what was in the gap
 - Thread call-stack backtrace samples
 - OS Runtime long function backtraces
 - API & library traces
 - User-coded annotations



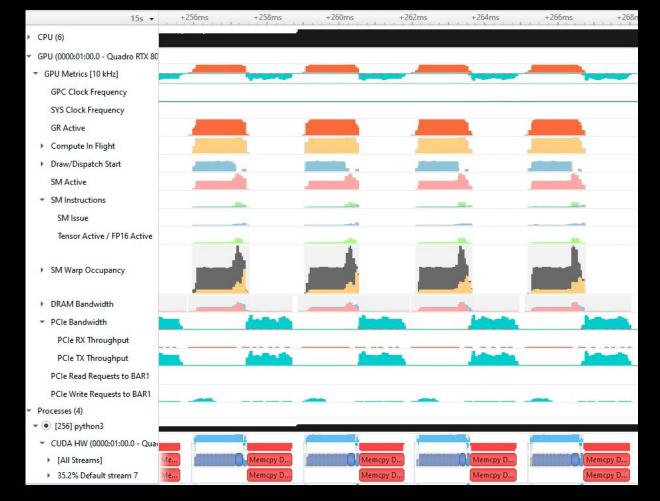
CUDA trace based GPU idle and low utilization level of detail Investigate by select kernels around bubble to find related CPU range

CUDA GPU utilization graph is based on percentage time coverage



Zooming in reveals gaps where there were valleys

CUDA streams eventually convert from graphs to ranges



GPU Metrics Sampling - how to interpret it

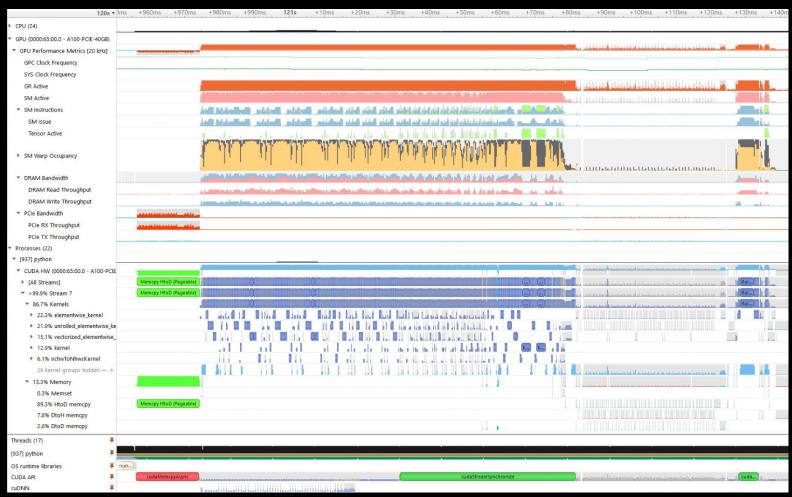
Interpreting GPU Metrics Sampling

- More info, no trace overhead, collected device-wide OOP
 - But no kernel names
- GR Active \Rightarrow It's doing some work
 - % GPU has any SM active (or NVENC, NVDEC, graphics)
- SM Active ⇒ How well is it using the width of the GPU
 - If low, increase batch sizes or look at kernel grid dimension
- SM Instructions Issued \Rightarrow Is it performing a lot of instructions
 - or might I be waiting on memory if low
 - Not enough warps to cover memory latency; larger kernel block dimensions can help
- SM Instructions tensor active ⇒ Using very faster special hardware
 - performance gains but slightly counter SM instructions can drop (vary by architecture)
 - can be limited by SM shared memory & waiting for loads
- Warp occupancy \Rightarrow Ratio of SM code types
 - Don't optimize for this! Ultra-optimized kernels don't always maximize warps!!! Ex: cuBLAS
- Memory and bus activity
- NOTE 1: Requires disabling DCGM and any DL framework built-in profiler
- NOTE 2: For lower priv access <u>https://developer.nvidia.com/ERR_NVGPUCTRPERM</u>

158s •)	lms +820ms +840ms	+860ms +880ms	+900ms	+920ms +940	lms +960ms +9	80ms 159s +20ms +40ms +60ms
CPU (24)						
GPU (0000:65:00.0 - A100-PCIE-40GB)						
GPU Performance Metrics [10 kHz]	1					
GPC Clock Frequency						
SYS Clock Frequency						
GR Active	. 1	t.		·		
SM Active		1 Contraction of the second se		14	In the second second second second second	
▼ SM Instructions	1	<u>e</u> .		1	فالطغالة الملجل المتعادية والتغالي	and the state of the second of the state of
SM Issue	1	. Provide the second		Provide La Constantina Constantina Constantina Constantina Constantina Constantina Constantina Constantina Const	. the second	and a state of the second s
Tensor Active		1			THE ALL ALL ALL STRANGS	ikili dala iki iki katala dala katala katala katala katala 🛶 🛶
SM Warp Occupancy						
 DRAM Bandwidth 		A		Distance in the local line	Line of the state of the state of the state of the	a a da cala da da da da da ana antia atra atra atra da da distrita da terra da terra da destructiva da como de
DRAM Read Throughput		. b		Charles and the second s	a loss and a loss and a loss of a second second	the standard state and the state and the state of the state
DRAM Write Throughput		la.		Street and a second sec	bellevil of a set a superior	and the second of the second
PCIe Bandwidth	A REAL PROPERTY AND A REAL PROPERTY AND A REAL PROPERTY.					
Processes (5)						
• • [757] python						
* CUDA HW (0000:65:00.0 - A100-PCI		1	i i i	11		
	Memcpy HtoD (Pageable)	conv2d_grouped_direct_kernel	precom explicit_c	Convolv		
[All Streams]						
▼ 95.8% Stream 13		conv2d_grouped_direct_kernel	precom explicit_c	Convolv		
99.8% Kernels	i	conv2d_grouped_direct_kernel	precom explicit_c	convolve		
▶ 0.2% Memory		11				
NVTX (tensorflow-core)	1	Conv2D: resnet50/conv1/C	onv2D 2889848113 [99.828 ms]			
11 streams hidden+	Memcpy HtoD (Pageable)					
 Threads (115) 						
- 🗸 [847] python -						
OS runtime libraries	pthrea					pthread_cond_wait
tensorflow-core	Send: IteratorGetNext/_11 2499	Conv2D: resnet50/conv1/Co	nv2D 2889848113 [99.369 ms]			
CUDA API	cuMemcpyHtoDAsync	cuEventSynchronize	cuEventSy cuEventSyn	cuEventSynchron]	S]. annumuna funtitään protontoisettiinist	
cuDNN						

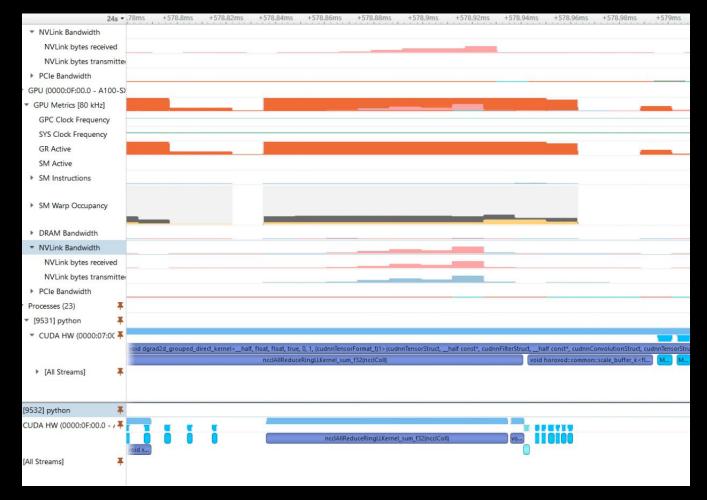
GPU Metrics Sampling Ex: TensorFlow2 ResNet50

O,



GPU Metrics Sampling - Mask-RCNN

NVIDIA



GPU Metrics Sampling - NCCL using NVLink

NVIDIA

Optimization Tactics (1)

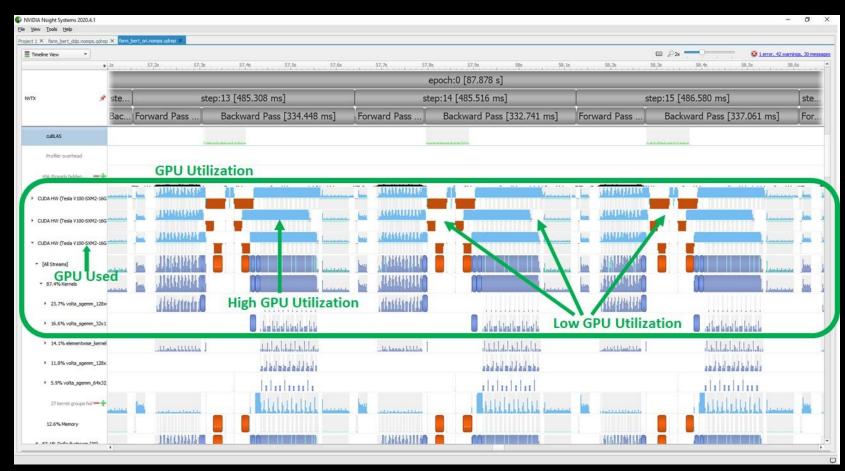
- Ensure using tensor cores & correct format to avoid conversions/transposes
- Increase batch/grid sizes to more efficiently use the GPUs' width
- Conventional parallelism
 - Increase framework's worker threads (CLI args)
 - Python async, await, future, tasks, Dask, ...
 - C/C++ OpenMP, OpenACC, pthreads, boost::asio, std::async, ...
- Parallel pipelining each stage can run in parallel
 - Are there a data dependencies between stages? No? Parallelize!
 - Ex: prefetch next batch/iteration of data while current batch/iteration is executing
 - Ex: within loader: net transfer, map, parse, load & transform, upload to GPU
- Reorder could i do that sooner?
 - Prefetch load data sooner & in parallel so it's there before needed
- Fusing tiny kernels, copies, or memsets, or use cudaGraphs
- Overlap training (or inference)
 - CUDA MPS to share contexts & avoid switch overhead
 - Possibly difficult to fix both into memory.

Optimization Tactics (2)

- Pass buffers by pointer avoid copies
- Multi-buffering don't make everyone wait on the same piece of memory
 - Often referred to as double or triple buffering; consider swap patterns
- Avoid returning data to CPU

. . .

- Avoid CPU pageable memory (prefer pinned / page-locked)
- Avoid unnecessary synchronizes
 - Avoid cuda*Synchronize functions, use cudaStreamWaitEvent instead
 - Avoid synchronous memory operation
 - Avoid CUDA default stream (if multi-stream)
- Pre-allocate memory, or use recycling tactics
- Minimize CUDA managed memory page faults on CPU & GPU (use prefetch)



AWS Blog: Deepset achieves a 3.9x speedup and 12.8x cost reduction for training NLP models by working with AWS and NVIDIA

Deepset achieves a 3.9x speedup and 12.8x cost reduction for training NLP models by working with AWS and NVIDIA

https://aws.amazon.com/blogs/machine-learning/deepset-achieves-a-3-9x-speedup-an d-12-8x-cost-reduction-for-training-nlp-models-by-working-with-aws-and-nvidia/

Cliff-notes

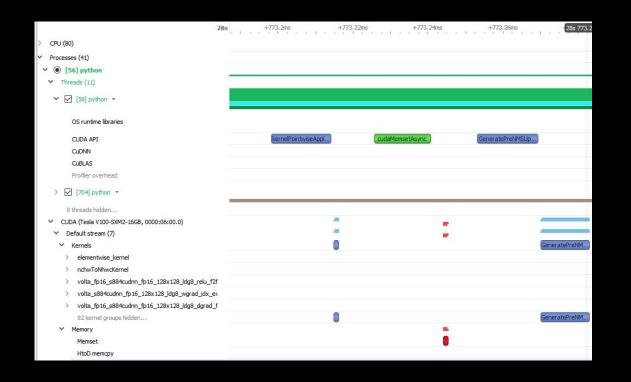
. . .

Heavy use of Nsight Systems

Switched from torch.nn.DataParallel (DP) to DistributedDataParallel (DDP)

Enabled larger batch sizes by switching to Automatic Mixed Precision (AMP)

Introduced a StreamingDataSilo & DALI to prefetch data



Fusion opportunities CPU launch cost + small GPU work size ≈ GPU sparse idle This can apply to DNN nodes/layers





cudaMemcpyAsync behaving synchronous Device to host pageable memory Mitigate with pinned memory

42

NVIDIA

	51s +197.395ms +197.4ms +197.405ms +197.41ms	+197,415ms +197,42ms +197,425ms +197,43ms +197,435ms	+197,44ms +197,445ms +197,45ms +197,455ms	+197,46ms +197,465ms +197,47ms +197,47 515 197,47817
CPU (12)				
cuDNN	cudnnBatchNormalizationForwardTrainin	3 [1		
CUDA API	bn_fw_tr_1C11_ker	cudaStreamSynchronize	cudaMemcpyAsync	cudaStreamSynchronize
Profiler overhead				
23 threads hidden				
CUDA (TITAN X (Pascal))				
> Stream 47				
✓ Stream 13				
✓ Memory				
Memset				
HtoD memcpy				
DtoH memcpy				
DtoD memcpy				Метсру
✓ Kernels		bn_fw_tr_1C11_kernel_new		

Example GPU idle caused by stream synchronization

43

CPU & OS

For cases not caused by CUDA API usage or clarified by NVTX



Permissions

Some features are tied to OS permissions

- CPU thread state, core occupancy, user-space call-stack periodic sampling
 - Paranoid level too high
 - Container SECCOMP blocking perf_event_open
 - OS kernel samples require even lower paranoid levels and/or sudo
- ftrace
 - CAP_SYS_ADMIN | CAP_SYS_PERFMON

See online Nsight Systems documentation, and UI warnings https://docs.nvidia.com/nsight-systems/InstallationGuide/index.html#linux-requirements https://docs.nvidia.com/nsight-systems/UserGuide/index.html#docker-profiling

OS Runtime API Trace

Example:Mask-RCNN

Map/unmap hiccups Mitigate by pipelining

- Map 1 batch ahead
- Unmap last batch
- Swap pointers here instead



	3s +437.9ms +437.95ms +438ms + <u>3s</u> 438.0614ms +438.1ms +438.15ms +438.2ms	+438.25ms +438.3ms +438.35ms +438.4ms +438.45ms +438.55ms +438.55ms
∽ 🔽 [12363] lm →		
OS runtime libraries	pthread	
NVTX	MPD_TI	VESTEP [760.124 µs]
INVIA		MPD_SYNCHRONIZE [548.501 µs]
CUDA API] c c [c] []]	cudaStreamSynchronize
✓ 🗹 [12364] lm →		
OS runtime libraries	pthread_mutex_lock	1
11.574	Duration: 6.818 μs	ESTEP [766.412 µs]
NVTX		MPD_SYNCHRONIZE [471.450 µs]
CUDA API	Call stack at 3.438s: cudaMemcpyAsync libpthread-2.23.so!_pthread_mutex_lock	cudaStreamSynchronize
✔ 🗹 [12366] lm 🖌	libcuda.so.384.81\027fa6033595f6 libcuda.so.384.81\027fa603359628 libcuda.so.384.81\027fa603359628	
OS runtime libraries	P pthread_mutex_lock libcuda.so.384.81!0x7fa60326c958 libcuda.so.384.81!cuMemcpyAsync	
NVTX	libcudart.so.9.0.176/0x7fa619dfd4fd libcudart.so.9.0.176/0x7fa619dda573	IMESTEP [780.810 µs]
INTA	libcudart.so.9.0.176/cudaMemcpyAsync Im!ZDivMultiGPUMapper:schedule_send()	MPD_SYNCHRONIZE [483.652 µs]
CUDA API	cudaMecc	cudaStreamSynchronize
✓ 🗹 [12357] lm →	libpthread-2.23.solstart_tread Libc-2.23.sol_clone	

OS runtime trace (OSRT) Includes backtraces of long running functions



NVIDIA

Choose functions		×	
Search criteria:	Check all Uncheck all		
Search		9	
<u>E</u> vents			
→ gk20a → gpio → hda → hda_controller → hda_centroller → huge_memory ↓ i2c ↓ imx185 ↓ ina3221 ↓ iommu ↓ ipi ✓ irq_handler_entry	Chocce functions		>
 ✓ irq_handler_exit ✓ softirq_entry ✓ softirq_exit 	<u>S</u> earch criteria:	<u>C</u> heck all	Uncheck all
✓ softirq_raise	irq		9
isomgr jbd2 kmem	Events		
Iibata Iibata migrate mipical mmc module			
	✓ softirq_exit ✓ softirq_exit ✓ softirq_raise ★ tegra_rtc tegra_rtc_irq_har	ndler	
		ОК	Cancel

45	+955.28ms +955.3ms +955.32ms +955.34ms
 CPU (6) 	
 Threads (5) 	
OS runtime libraries	sc sc sc sc sched_yield sc sc sc
CUDA API	cudaEve
Profiler overhead	
4 threads hidden 🛛 📥 🛖	
 CUDA (NVIDIA Tegra X2) 	
 FTrace events 	
softirq_raise	
softirq_exit	
softirq_entry	
irq_handler_exit	
irq_handler_entry	

FTrace Example demonstrates interrupts

Bottom-Up View

Filter... 99.82% (23,260 samples) of data is shown due to applied filters.

Symbol Name	Self, %	Module Name
VolumetricData::compute_volume_gradient()	20.14	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ VolumetricData::compute_volume_gradient()	20.14	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
BaseMolecule::add_volume_data(char const*, double const*, double const*, double const*, double const*, int, int, int, float*)	18.30	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
VMDApp::molecule_add_volumetric(int, char const*, double const*, double const*, double const*, double const*, int, int, int, float*)	18.30	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ obj_segmentation(void*, Tcl_Interp*, int, Tcl_Obj* const*)	18.30	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
[Max depth]	18.30	[Max depth]
BaseMolecule::add_volume_data(char const*, float const*, float const*, float const*, float const*, int, int, int, float*, float*, float*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
MolFilePlugin::read_volumetric(Molecule*, int, int const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
VMDApp::molecule_load(int, char const*, char const*, FileSpec const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ text_cmd_mol(void*, Tcl_Interp*, int, char const**)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ TclInvokeStringCommand	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ TclEvalObjvInternal	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ TclExecuteByteCode	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ TclCompEvalObj	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ TclEvalObjEx	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ Tcl_RecordAndEvalObj	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ TclTextInterp::evalFile(char const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ VMDApp::logfile_read(char const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
✓ VMDreadStartup(VMDApp*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
[Max depth]	1.84	[Max depth]
> 0x7f10ca7022d6	5.13	/usr/lib64/libcuda.so.390.25
> obj_segmentation(void*, Tcl_Interp*, int, Tcl_Obj* const*)	3.44	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11

Function table shows statistics from periodic call-stack backtraces

49

CLI statistics and export



DVIDIA

DL Rank Data Processing Imbalances

- Hurry up and wait
 - If anyone takes longer to reach the all-reduce then everyone is stuck!!!
- Load time
 - Did this batch take longer than the norm to load?
 - If parallel, did any rank have to wait?
- Processing time
 - Did this rank take longer than other
- Some remedies
 - Fix the data
 - 1 JPEG among PNGs or 1 MP3 among WAVs?
 - Wrong resolution, sample rate, precision, …
 - Reorganize your batch order
- A perfect job for scripts & statistics



Time(%)	Time (ns)	Calls	Avg (ns)	Min (ns)	Max (ns)	Name
13.6	17198585879	66612	258190.5	20864	1327579	nhwc_batch_norm_fwd
11.3	14228434514	20075	708763.9	168352	1786335	nhwc_batch_norm_bwd_add_relu
10.4	13100639465	40158	326227.4	177824	1342749	volta_fp16_s884cudnn_fp16_128x128_ldg8_dgrad_f2f_exp_interior_nhwc_tt_v1
10.1	12688128728	42745	296833.1	168895	760641	volta_fp16_s884cudnn_fp16_128x128_ldg8_relu_f2f_exp_interior_nhwc_tn_v1
9.3	11744819167	41418	283568.0	167680	753631	volta_s884cudnn_fp16_128x128_ldg8_wgrad_idx_exp_interior_nhwc_nt
7.1	8935340081	23848	374678.8	229056	895710	volta_s884cudnn_fp16_64x64_sliced1x4_ldg8_wgrad_idx_exp_interior_nhwc_nt
6.2	7829371510	41408	189078.7	28800	1365372	nhwc batch norm bwd relu
4.1	5145597181	13820	372329.8	325313	524159	volta_fp16_s884cudnn_fp16_128x128_ldg8_relu_f2f_exp_small_nhwc_tn_v1
3.8	4796840892	12551	382187.9	311649	775264	volta fp16 s884cudnn fp16 128x128 ldg8 dgrad f2f exp small nhwc tt v1
2.9	3599110831	482226	7463.5	1760	868095	dctQuantInvJpegKernel
2.4	2968987604	5018	591667.5	124767	1418268	nhwc batch norm bwd
2.0	2550050661	162488	15693.8	1184	1720191	ycbcr to format kernel roi
1.6	2031335889	3768	539101.9	381407	784829	dgrad 1x1 stride 2x2
1.5	1843345484	1256	1467631.8	1451800	1691492	pooling_bw_max_nhwc_kernel
1.2	1565496722	3781	414043.0	391712	480321	volta fp16 s884cudnn fp16 256x64 ldg8 relu f2f exp small nhwc tn v1
1.2	1543111896	3765	409857.1	386239	2808775	volta fp16 s884cudnn fp16 256x64 ldg8 dgrad f2f exp small nhwc tt v1
1.1	1373574502	2512	546805.1	434785	760834	dgrad 1d
1.0	1267659803	2514	504240.2	492895	601119	volta fp16 s884cudnn fp16 256x128 ldg8 relu f2f exp small nhwc tn v1
1.0	1215018985	58722	20691.0	1152	1325441	MapPlanKernel
0.8	1020510778	5670	179984.3	960	737985	mxnet generic kernel
0.7	909701850	65290	13933.2	6400	123808	convertTensor_kernel
0.7	894204599	2536	352604.3	192223	3158098	BatchedSeparableResampleKernel
0.7	868771063	1257	691146.4	670495	821372	first_layer_wgrad_kernel
0.7	859121367	1261	681301.6	675584	793889	first layer fwd kernel
0.7	854624303	1257	679892.0	661086	726018	volta_fp16_s884cudnn_fp16_256x128_ldg8_relu_f2f_exp_interior_nhwc_tn_v1
0.6	719945062	1256	573204.7	555264	648066	dgrad 2d
0.4	479525740	6430	74576.3	2560	1835937	dcAcDecodeKernel
0.4	450010850	1268	354898.1	223551	1985854	BatchedCropMirrorNormalizePermuteKernel
0.3	436136496	1260	346140.1	342111	361025	pooling_fw_kernel_max_nhwc
0.3	364887793	1257	290284.6	164992	395295	volta_fp16_s884cudnn_fp16_256x64_ldg8_dgrad_f2f_exp_interior_nhwc_tt_v1
0.3	354605440	6430	55148.6	2176	1460961	transposeKernel
0.3	336891222	92	3661861.1	892835	26052414	convolve_common_engine_float_NHWC
0.2	302099259	65266	4628.7	928	48576	scalePackedTensor_kernel
0.2	277514711	123110	2254.2	1216	34304	computeOffsetsKernel
0.2	194972504	1261	154617.4	150848	177536	volta_fp16_s884cudnn_fp16_256x64_ldg8_relu_f2f_exp_interior_nhwc_tn_v1
0.1	176725421	46	3841857.0	578114	7942931	wgrad_alg0_engine_NHWC
0.1	160091745	65268	2452.8	1599	20384	computeWgradOffsetsKernel
0.1	116475012	1256	92734.9	91520	242912	pooling_bw_avg_nhwc_kernel
0.1	115380220	2159	53441.5	3008	1234110	destuffKernel
0.1	108445324	57731	1878.5	960	34016	computeBOffsetsKernel
0.1	77051216	23	3350052.9	1496867	7849586	wgrad_alg1_engine_NHWC
0.1	64827363	1256	51614.1	49663	67488	pooling fw 4d kernel

Stats/Export - CUDA kernel summary



Time(%)	Time (ns)	Instances	Avg (ns)	Min (ns)	Max (ns)	Range
4.0	162789746	70	2325567.8	248462	10421312	TensorRT:ExecutionContext::enqueue
0.4	16335803	24	680658.5	94707	2036983	TensorRT:predictions
0.3	10447691	70	149252.7	16094	3604518	TensorRT:conv1 + activation 1/Relu
0.2	10169055	70	145272.2	25936	1169143	TensorRT:conv1 + activation_1/Relu input reformatter 0
0.2	8698016	24	362417.3	9581	2970733	TensorRT:block 3b conv 2
0.2	8625565	24	359398.5	28257	1242590	TensorRT:average pooling2d 1
0.2	7733550	46	168120.7	30474	3498224	TensorRT:conv2d cov/Sigmoid
0.2	6672310	70	95318.7	11271	1151050	TensorRT:block 1a conv 1 + activation 2/Relu
0.1	5312256	70	75889.4	8316	2133255	TensorRT:block_4a_conv_2
0.1	5197452	24	216560.5	9572	1543570	TensorRT:block_3b_conv_1 + activation_12/Relu
0.1	4922521	70	70321.7	11732	1281643	TensorRT:block 1a conv shortcut + add 1 + activation 3/Relu
0.1	4151987	24	172999.5	10592	1625630	TensorRT:block_1b_conv_2
0.1	4121931	70	58884.7	10616	1307464	TensorRT:block_2a_conv_2
0.1	4108970	70	58699.6	9311	834374	TensorRT:block_1a_conv_2
0.1	4027181	24	167799.2	10564	1728250	TensorRT:block_1b_conv_shortcut + add_2 + activation_5/Relu
0.1	4018713	24	167446.4	9636	2431179	TensorRT:block_2b_conv_1 + activation_8/Relu
0.1	3799363	70	54276.6	8595	481048	TensorRT:block_3a_conv_2
0.1	3762886	24	156786.9	13969	1008096	TensorRT:block_4b_conv_shortcut + add_8 + activation_17/Relu
0.1	3376998	24	140708.2	12773	1641372	TensorRT:block_4b_conv_2
0.1	3361499	24	140062.5	11195	1248129	TensorRT:block_2b_conv_2
0.1	3227736	24	134489.0	15949	736287	TensorRT:predictions/Softmax
0.1	3023238	24	125968.2	11972	827474	TensorRT:block_2a_conv_shortcut + add_3 + activation_7/Relu
0.1	2998494	24	124937.2	12327	1217984	TensorRT:block_2a_conv_1 + activation_6/Relu
0.1	2853998	46	62043.4	10971	1983123	TensorRT:block_3a_conv_shortcut + add_3 + activation_7/Relu
0.1	2773491	24	115562.1	12019	809395	TensorRT:block_4a_conv_shortcut + add_7 + activation_15/Relu
0.1	2744711	24	114363.0	10474	892787	TensorRT:block_2b_conv_shortcut + add_4 + activation_9/Relu
0.1	2147200	24	89466.7	12037	645408	TensorRT:block_3a_conv_1 + activation_10/Relu
0.0	1914194	24	79758.1	10599	842947	TensorRT:block_4b_conv_1 + activation_16/Relu
0.0	1732011	24	72167.1	9659	488562	TensorRT:block_3a_conv_shortcut + add_5 + activation_11/Relu
0.0	1675232	24	69801.3	10073	623120	TensorRT:block_4a_conv_1 + activation_14/Relu
0.0	1525283	24	63553.5	10386	540733	TensorRT:block_3b_conv_shortcut + add_6 + activation_13/Relu
0.0	1358781	46	29538.7	10645	241099	TensorRT:block_3a_conv_1 + activation_6/Relu
0.0	1301126	46	28285.3	11929	244583	TensorRT:conv2d_cov

Stats/Export - NVTX code annotations Note this includes TensorRT domains

2s +276ms +277ms +278ms +279ms +280ms +281ms +282	ms +283ms +284ms +285ms
Threads (1) 958 pc 0 msec	
▼ ▼ [3400] ▼ 958 px; 0 msec	
DX12 API 958 px: 0 msec ID3D12CommandQue ID	ID3D12Device::CreateCommitte, ID3D12 ID3D12G
Scene Render Command List Markers 958 pc 0 msec Particle Up Rend Z Pr Ge Pa	Post Effects HDR Tone Mapping
O Re C	
Profiler overhead 958 p DX12 profiling o	
▼ Frame duration (60 FPS) 958 pc; 0 msed	
► CPU frame duration 958 px; 0 msec	-
► GPU frame duration 958 px; 0 msec	
▼ DX12 958 pc 0 msec	
	ics command list [Reset,Close]
Command Lists Creation 958 px; u msec	Graphics
359 Swap Chain 0 958 px 0 msed	
4	Þ
Events View V Pr cess [16084] ModelViewer.exe (1 of 1 thread)	
Filter All	Search
# Name Duration TID Start 686 ID3D12GraphicsCommandList:BeginEvent 5.900 μs 3400 2.27697s	CPU Marker Start (5) Opaque
687 → CPU Marker Start (5) Particle Undate 200 ns 3400 2,277s	Start: 2.27812s
690 ID3D12GraphicsCommandListEndEvent 200 ns 3400 2.27765s	End: 2.27812s
691 CPU Marker Start (5) RenderLightShadows 200 ns 3400 2.27768s	Duration: 1 ns Thread: 3400
694 ID3D12GraphicsCommandListEndEvent 100 ns 3400 2.27807s	
695 v CPU Marker Start (5) Z PrePass 200 ns 3400 2.2781s	
696 ID3D12GraphicsCommandList::BeginEvent 200 ns 3400 2.2781s	
1 697 ▼ CPU Marker Start (5) Opaque 1 ns 3400 2.27812s	
ID3012GraphicsCommandList::BeginEvent 1 ns 3400 2.27812s	
I 699 CPU Marker End (5) 200 ns 3400 2.27836s	
700 ID3D12GraphicsCommandList:EndEvent 200 ns 3400 2.27836s	
1 701	
704 ID3D12GraphicsCommandList::EndEvent 100 ns 3400 2.27842s	
705 CPU Marker End (5) 100 ns 3400 2.27843s	
706 ID3D12GraphicsCommandList:EndEvent 100 ns 3400 2.27843s	

Event Table & Statistics Table





Multi-node

All the problems on a single node server (DONE)

Now you need to worry about

Working with your cluster job/task scheduler

Multi-report views

But how do you pick which reports among thousands?

Wall-clock time

Networking

Data analysis

Working with your cluster work scheduler

- srun <slurm_args> nsys profile <nsys_args> app <app_args>
- Make sure your report names are unique
 - --output=friendlyName_%q{SLURM_NODEID}_%q{SLURM_PROCID}.nsys-rep
- Avoid getting system-wide data from all ranks by moving nsys & app into a shell script
 - IF [\$SLURM_LOCALID == 0] THEN
 - Add --nic-metrics=true
 - Add --gpu-metrics-device=all
- LIMIT your recording time otherwise you will run out of memory opening reports in the GUI on your laptop
 - 30sec-1min is often good enough. Or even less... capture a few iteration.
- I have more reports than I know what to do with. Now what?

<u>File V</u> iew <u>T</u> ools <u>H</u> elp			
<u>N</u> ew Project	Ctrl+N		
<u>O</u> pen	Ctrl+O		① 17 messages
Add Report (beta)	Ctrl+T		8s A
Import	Ctrl+I	2s 3s 4s 5s 6s 7s	
Export jacobi_2mpi.qdrep [2 rep	orts] Ctrl+E		
<u>Close jacobi_2mpi.qdrep [2 repo</u>	rts] Ctrl+W		
Exit	Ctrl+Q		
1011	in a conservation of the c	Jacobi solve [3,472 s] Jacobi solve [1,811 s]	
NVTX	1179 px; 0 msec		
CUDA API	1179 px; 0 m cuda cudaHos	🛄 cudaHo)	
Profiler overhead	117 k; 0 msec		1
9 threads hidden – +	117 <u>9 px; 0 msec</u>		
CUDA HW (0000:06:00.0 - Tesla	1179 px; 0 msec		
 prm-dgx-16 (0:0) 	1179 px; 0 msec		
▶ CPU (80)	1179 px; 0 msec		
✓ Threads (10)	1179 px; 0 msec		
💌 🗸 [19265] MPI Rank 0 🕞	and the second second		
OS runtime libraries	11 ioctl	lioctl	
MPI	11 [MPI_Init]sec		
NVTX	1179 px; 0 msec	Jacobi solve [3,474 s] Jacobi solve [1,811 s]	
	1179 px; 0 m cuda cudaHo.		
2.841 CUDA API			

Loading Multiple Reports into one Timeline

<= 2022.1 via file menu
>= 2022.2 via open dialog, select multiple report
NOTE: Pick a subset of reports otherwise you may run out of RAM

Wall-Clock Time

Timelines need it, especially if they are going to fit together accurately

Data is collected independently

Relies on the system's wall clock time to be accurate

Otherwise there is timeline drift

From worst to best

- NTP = 1ms accuracy
- PTP software = 10 to 100 microseconds accuracy
- PTP hardware = can get down to ~10ns

Data Analysis

- Did not manually look at all files.... too many!
 - Maybe mix-n-match randomly with multi-report views?
- Use data analysis to hone in on interesting iterations \rightarrow ranks \rightarrow report
 - Cluster iteration times global total time
 - Per-rank iteration data load times
 - Per-rank iteration processing times
 - Total time to reach all-reduce
 - Forward pass
 - Backward pass
 - Per-rank iteration communication time, if not overlapped
 - DNN layer stats?
- Visually compare an average, min, & max report to try to understand how and why they differ
- You may need to add NVTX to your app to get some of this information
- GTC Talk: Scaling Transformer in PyTorch Across Multiple Nodes
 - <u>https://developer.nvidia.com/gtc/2020/video/s21351</u>
 - We'll make this type of stuff easier in the future

Data Processing Imbalances

- Make every rank consistent in the iteration
- Load time
 - Did this batch take longer than the norm to load?
 - If parallel, did any rank have to wait?
- Processing time
 - Did this batch take longer than the norm to process with the DNN?
- Hurry up and wait
 - If anyone takes longer to reach the all-reduce, everyone is stuck!!!
- Some remedies
 - Fix the data
 - A lonely JPEG in a PNG world, or MP3 among WAVs?
 - Wrong resolution, sample rate, precision, ...
 - Reorganize your batch order

Other Products



Nsight Compute CUDA Kernel Profiler

Interactive CUDA API debugger

Advanced CUDA Kernel Profiling

CUDA-C/PTX/SASS correlation

Source correlated performance metrics

Diff'ing for performance reports

Programmable expert system

NVTX-range-defined kernel profiling

High performance GUI visualization and CLI data collection

NOTE: See earlier slides about relationship with Nsight Systems! Start there to get big picture!

Windows 10, Linux Ubuntu 16.04/18.04/20.4, RHEL 7.x

			×	out reight-caprel report					
				Page Source - Process A			max - Add Baseline - Appl		Copy as Inte
ext Tripper: Enter reges									
				Current 16052 - pooling_br	e_5d_kernel_max(102 Time: 3	06 miscond Cycles: 3	35,291 Regs: 31 GPU: Xavier	SM Prequency: 114	4.155.093.05 CC: 7.2 Process: (12239) cud
ID API Name	Details		Func Parameter *	View SASS +					
16307 cudaGetLastError 16305 cudaLaurchKernel		cudaSaccess(0) cudaSaccess(0)	(8-7(6)-97170						a 55 58 55 PS6 18cudrePoolnaStructT0 58 i
16309 CeLasschRernel		CLEA SUCCESS(0)						enserStructPRT_SS_S	1_SS_SB_SS_PS6_18cudnePcolingStructT0_SB_1
16310 gen segamend	ore sequenced								
				Address Sa	sr(+	Sampling Data (ATI	Sampling Data (Not Issued) In	tructions Executed	Predicated On Thread Instructions Executed
				105 09900002 11265310	1940.wite no. 015, 00, close			1,21	41.500
16313 cstaunchKernel 16314 cealacettans kerne		CLEA_SUCCESS(0)	(0.e4180e200, 2					5,916	115,531
16315 cudaGetLastError		cudaSuccess(0)		168 0000002 11265330	ISETP.6E.AND P2. PT. R15. R			B1816	113.530
16315 CUBACHTLASTEFFOF		cudidaccess(d)	(0+42/4/2250 0	109 0000002 11265340				1000	115,530
16317 cellenceu0toH v2		CLEA SUCCESS(0)		110 0000002 11265350 022 111 00000002 11265360	0011265470 100.6.64.575 84, [86]			13181.6	115,531
16318 cudaFree				112 00000002 11265370	INAD R17, R16, c[0x0][0x270]			0.010	113,530
		CLEA_SUCCESS(0)		113 0000002 11265380	IMAD. NOV. 032 824, 82, 82, 8			0,816	113,531
		cudaSuccess(0)			SHFL.10X PT. NZ, NZ, NZ, NZ		ż (11,448	333,179
16321 cuflenFree_v2 16322 cudaFree		CUDA_SUCCESS(0) cudeSuccess(0)			1940-904-012 R3, RZ, RZ, 04		•	11,440	333,179
16323 cutientree v2		CLEA SUCCESS(0)		116 0000002 11265380 117 00000002 112653c0	1940 R18, R24, c[000][0x27c] 1940.9000 R2, R18, R3, c[00		0	11.448	333,179
16324 cudaFree				110 00000002 11265360	LOS.6.64.5Y5 R2, [R2]		4	11,448	333,178
16325 cutienFree_v2		CLEA_SUCCESS(0)		119 00000002 112653e0	855Y 84 0(21)265488		2	11,448	333,179
				120 0000002 112653/0	DEETP. DTU. AND P2. PT. 1021.	2.179	1.772	11.440	333,179
		CLEA_SUCCESS(0)				12		11,448	333,179
16328 cudaFree 16329 culterFree v2		cudaSuccess(0) CUDA SUCCESS(0)			DSETP.ETU.AND P2, PT, [84].			2.87	
16330 cudaNencov		cudeSuccess(0)			1NAD.NOV.032 825, 82, 82, 0			2, 🌆	
16331 cumencevitto0 v2		CLEM. SUCCESS(0)		124 0000002 11265430 125 00000002 11265440 002	PENT R25, R25, 0x7610, R25 5% 0x211285470			2.87 2.87	3,484 3,484
16332 cudaNencpy					DEETP.NEU_AND P2. PT. R2. R	100		11.440	3,404
				127 0000002 11265460	SEL R25, R2, (K), P2	198	188	11,448	329,669
				128 09999902 11265478	85116 84		- a	13,635	333,179
16335 cullencpyRtoD_v2 16326 cudaMencpy		CUDA_SUCCESS(0) cudaSuccess(0)		129 0000002 11265480			4 (11,448	333,179
16330 cutoMencpy 16337 cuttencpy#to0 v2		CLEAS SUCCESS(0)		130 0000002 11265490			3 [11,448	333,179
16335 cudaLaunchKernel			(0x7f701ae488.	131 0000002 112654x0 072 132 00000002 112654x0 072	DREAK D3		-	11,440	335,178
16339 cetaeschkernel			(0x3c79ac80, 4	132 0000002 11265420 02	14003 R24, R24, 0x1, R2		2	11,448	319,381
				134 0000002 11265460	ISETP.GE.AND P2. PT. R24. R		1	11,440	319,301
							9	11,448	319,381
								57616	227755
								137,916	99,755
				138 0000002 11265510 139 00000002 11265520 0192	ISETP.6E.AND P2. PT. R16, R			37816 57816	99,733
				159 0000002 11265520	1940.00211205520 1940.009.022 818, 82, 82, -1			1,223	20,733
				141 00000002 11265540	BSING R2			5,350	4,500
				142 0000002 11265550	ISETP.CE.OR. P2. PT. 121. cl			1,212	-0.500
				143 0000002 11265560	RSSY 82, 00/211265818			1,21	4,500
				144 00000002 11265570	IMAD.MOV.032 817, 82, 82, -			1,273	-
				145 0000002 11265500 (92)				1,22	4.533
				147 00000002 11265540	14603 R6, R13, c[0:0][0:20c] 1940.NDV.032 R7, R2, R2, 0c			1,22	01,500 a 1990
				140 00000002 11265560	1/403 R25, R23, c10x0110x271			1,200	-1.200
				149 00000002 112655c0	DMD.NOV.032 816, 82, 82, 8			1.28	41,500
				150 0000002 11265540	INAD.WIDE NO. NO. N7. c[0x0]			1,27	40,000
					SHFL.IDX PT, RZ, RZ, RZ, RZ			5,816	115,600
				152 0000002 11265510	ISETP. CE. AND P2. PT. R15, R			131816	113,606

Nsight Deep Learning Designer

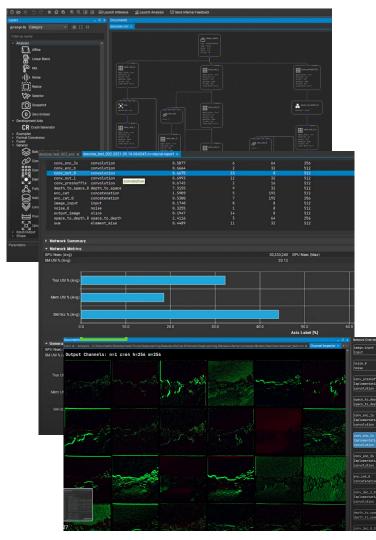
A new tools to add DL-based image processing feature to applications that have strict performance requirements

Designer, inspector, profiler image & video processing models

Download https://developer.nvidia.com/nsight-dl-designer

Quick intro(2.5min) <u>https://www.youtube.com/watch?v=7AraPM8dhyc</u>

Extended Intro (35min) https://www.nvidia.com/en-us/on-demand/session/siggraph2021-s igg21-s-05/?playlistId=playList-83726b46-d20a-40c2-a218-18c1b 5fb0759



Nsight Visual Studio Code Edition

VSCode plugin for CUDA dev, compile, & debug

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

Quick Intro (4min)

https://www.youtube.com/watch?v=gN3XeFwZ4ng

Microsoft Intro (14min)

https://www.youtube.com/watch?v=I6PgYhiQr-I

<pre>// Performs warmup operation using matrixMul CUDA if (block size == 16)</pre>	kernel
<pre>{</pre>	
MatrixMulCUDA<16>	
<pre></pre>	B, dimsA.x, dims
}	
else	void MatrixMul
1	float *B, int
MatrixMulCUDA<32><< <grid, 0,="" stream="" threads,="">>></grid,>	-(b)
}	
- A state in the second in	
<pre>printf("done\n");</pre>	
<pre>checkCudaErrors(cudaStreamSynchronize(stream));</pre>	
// Record the start event	
<pre>checkCudaErrors(cudaEventRecord(start, stream));</pre>	

File E	dit Selection View Go Run Termina	Help								
Ð	RUN AND DE 🕨 CUDA: Debug 🗸 🕲 😶	• • • • • • • • • • • • • • • • • • •	.cu X						Э	
3	\sim VARIABLES	G- matrixMu	ıl.cu > 😚 MatrixMulCUI	DA(float *,	float			nt, int		
Q 2 2 2 2 1 2 2 1 2 1 2 1 2 1 2 1 2 1 2	✓ Local k: 0 > 8s: 0x1000 > As: 0x0 a: 0 b: a ✓ CALL STACK ✓ (CUDA) PAUSED ON BREAKPOI MatrixMulCuDA-32> matrixMul.cu 1 ⁺	n 116 117	<pre>// Synchronize syncthreads() // Multiply the // each thread // of the block agma unroll for (int k = 0; {</pre>	; two ma compute sub-ma		tes t ne el	toget Lemer			
¢.	> matrixMul PAUS > matrixMul PAUS > cuda-EvtHandir PAUS > matrixMul PAUS	D 120 D 121	<pre>Csub += As[ty // Synchronize // computation // sub-matricessyncthreads() // Write the bloc</pre>	to make is done of A a ;	sur bef nd E				two t it	

ADDITIONAL CUDA DEVELOPER TOOLS

Command-line CUDA tools

CUDA-gdb

Unified CPU and CUDA Debugging

CUDA-C/PTX/SASS support

Compute Sanitizer - API and utility

memcheck : reports out of bounds/misaligned memory access errors

racecheck : identifies races on __shared__ memory

initcheck : usage of uninitialized global memory

synccheck : identify invalid usage of __syncthreads() and __syncwarp()

THANK YOU!

Downloadhttps://developer.nvidia.com/nsight-systemsNOTE:Website version is newer than CUDA Toolkit

Let us know about your successes!