

Nsight Compute 41TH VI-HPS TUNING WORKSHOP

Felix Schmitt

Nsight Product Family

Nsight Systems - Analyze application algorithms system-wide

Nsight Compute - Analyze CUDA kernels

Nsight Graphics - Debug/analyze graphics workloads

Workflow





Overview

				D Untitled 1 * ×								
NEL N:4[SRC] G:0-+0[SRC]				Paper Details - 1	inch: 0.: 654.00	at the later		Add Danalis	a w			Corrie an Image
				Logic Decision - Decision	anch of our re	Time	Ord	Reas GPL		SM Frequency	CC Process	
				Current 65	- reduce (612 1 1	h(512 1_ 234.7	2 usecond 318	783 16 NVI	DIA GeForce RTX 2080	TI 135 cycle/nsecon	d 7.5 [4969] simpleDataGr	inha la
	Details											
cuDeviceGetAttribute		CUDA_SUCCESS(0)	(0x7ffee29ef39c(1), CU_DEVICE_AT	GPU Speed Of Light 1	hroughout							0
cuCtxGetCurrent		CUDA_SUCCESS(0	(0x7ffee29ef3a0(0x55869169c630)	Compute (SM) Through	ut [%]			78.98	Duration [usecond]			234.72
cuuraphAddMemcpyNode faGranhGetNodes		cudaSuccess(0)	(0x558691ce6258.0x0.0x7ttee29e	Memory Throughput [%]				45.63	Elapsed Cycles cycle			318,783
cuGraphGetNodes		CUDA_SUCCESS(0)	(0x558691ce6258.0x0.0x7ffee29e	L1/TEX Cache Throughp	ut [%]			10.34	SM Active Cycles (cyc	ie]		289,849.71
taGraphinstantiate		cudaSuccess(0)	(0x7ffee29ef678(0x558691ceeb88)	L2 Cache Throughput [%				14.54	SM Frequency (cycle/	nsecond]		
uGraphinstantiate_v2		CUDA_SUCCESS(0)	(0x7ffee29ef678(0x558691ceeb88)	DRAM Throughput [%]				45.63	DRAM Frequency (cyc	le/nsecond]		6.76
aGraphCione		cudaSuccess(0)	(0x7ffee29ef680(0x558691cf9b38)	A High Compute Th	compa	ute is more heavily	utilized than Mer	nory: Look at the			to see what the compute pipeli	nes are
utraphicione		CUDA_SUCCESS(0)	(0x7ffee29ef680(0x558691cf9b38)	- Angel Computer II	spendi	ing their time doing	a Also, consider 1	whether any corr	putation is redundant	and could be reduced o	or moved to look-up tables.	
uGraphinstantiate v2		CLIDA SUCCESS(#	(0x7ffee29ef690/0x558691cfd1c8)			and Bank Health and	And Add States	formania and	And the second second		a designed for the second second	
laGraphLaunch			(0x558691ceeb88, 0x558691cdd50	A FP64/32 Utilizati	an its fp64 peak p	performance. If	uccone (1po4) per	determ	ines that this kernel is	fp64 bound, consider u	using 32-bit precision floating p	icie and 19% of
uGraphLaunch			(0x558691ceeb88, 0x558691cdd50		to improve its	performance.						
MEMSET N:3[SRC] G.O.		cudaSuccess(0)	dst: 0x7f8b9d601000, pitch: 0, val: (Compute Workload A	nelvels							
MEMSET N:1[SRC] 0.0-		cudaSuccess(0)	dst: 0x7f8b9d600000, pitch: 0, val: 0	Executed tpc Elapsed lin	st/cycle]			0.30	SM Busy [%]			86.65
MEMCPY NUISRC GO	in second	cudaSuccess(0)	Host ox/repsedoboool(o,o,o) LOD: 0	Executed Ipc Active [inst	/cycle]			0.33	Issue Slots Busy [%]			
KERNEL NULSBOOK 0:0->	reduceFinal	cosasuccess(0)	orid: 1 x 1 x 1 block: 512 x 1 x 1,	Issued Ipc Active [inst/c	(cte)			0.33				
and a second second second			E State of the sta		EPG4 is the h	inhest attitud site	eline (86.6%) It a	and the second second	nation point operations	The singline is merur	tilized and likely a performance	
				Very High Utilized	on bottleneck.	A stranged bib			Press of Paralleline			
				· Manual Moderat Av	abata .							
				Memory Throughout (Ch	again the second			296 10	Mars Bary Ph.I			14.53
				L1/TEX HIT Rate [%]				0.00	Max Bandwidth [%]			45.63
Cail ID CUstream	Stream ID	CUcontext	Priority Access Policy Wind: *	L2 Hit Rate [%]				0.16	Mem Pipes Busy [%]			
	0x0			a manufacture constraints								
	0x0			Active Warms Der Schart	the Designal			12.24	tes Eticible In1			
	0x0		0 unset	Fligible Warps Per Sched	ister Iwarpl			0.10	One or More Eligible I			8.34
	0x0		0 unset	Issued Warp Per Schedu	kr			0.08				
601	0x0	10 00000000000000	o unset									
601	0x0		0 Unset		Every schedu leave hardwa	aler is capable of it are resources unde	isoing one instru rutilized and may	ction per cycle, b r lead to less opt	ut for this kirnel each imal performance. Out	scheduler only issues a of the maximum of 8 v	an instruction every 12.0 cycles warps per schedulet, this kerne	. This might I allocates an
601	0x0		0 unset	A Issue Slot Utilizat	ion average of 7	36 active warps p	er scheduler, but	only an average i	of 0.10 warps were elk	tible per cycle. Eligible	warps are the subset of active	warps that are
614 0+559601+44	1600	THE OWNER AND A DESCRIPTION OF	o unset		the number of	of eligible warps, a	void possible loar	d imbalances du	e to highly different ex	ecution durations per w	varp. Reducing stalls indicated	on the winth
644	040	15 .0075580110000		No. of Concession, Name		Concession of the local division of the loca					WINDOW CONTRACTOR	and the second second
644	0x0	16 Decompositores	GPUBIackSchol	escallPut(in	t, float	** +10at		10°, +10	oat*, +100	it*), 2019	-Aug-12 14:44	erse, conte
644	0x0		Section: GP	U Speed OF L	ight							
			Nemory Freq	uency								Shiz
is into API Statistics	NVIX		SOL FB									× .
ead Divergence" (0 new r			Elanced Cur	145							(esti	-le
			Cit Economic	2								City of
			ser rrequenc	7								arra ar
			Presory [%]									
			Duration								usec	ond
			50L L2									1.2
			SN Active C	vcles							EV.	tle
			SH [12]									*
			CAL TEX									1000
			SOF LEX									-

Interactive CUDA Kernel profiler

Targeted metric sections for various performance aspects

Customizable data collection and presentation (tables, charts, ...)

UI and Command Line

Python-based API for guided analysis and post-processing

Support for remote profiling across machines and platforms

Profiling Activities

Interactive Profile

5341 > de	evice_tea_leat	_ppcg_solve_update_r 👻				Export to CSV
xt Trigger	vice_tea_le	af_ppcg_solve_(calc update	e).* ×			
	API Name		Details		Func Return	Func Parameter 👌
43833	cudaMalloc				cudaSuccess(0)	(0x555d75b64830{(
43834	cuMemA	lloc_v2			CUDA_SUCCESS(0)	(0x555d75b64830{(
43835	cudaMalloc				cudaSuccess(0)	(0x555d75b64838{(
43836	cuMemA	lloc_v2			CUDA_SUCCESS(0)	(0x555d75b64838{(
43837	cudaMemor	y v			cudaSuccess(0)	(0x7feb89fecc00, 0x
43838	cuMemc	pyHtoD_v2			CUDA_SUCCESS(0)	(0x7feb89fecc00, 0x
43839	cudaMemor	y v			cudaSuccess(0)	(0x7feb89fed400, 0>
43840	cuMemc	yHtoD_v2			CUDA_SUCCESS(0)	(0x7feb89fed400, 0)
43841	cudaLaunch	rKernel			cudaSuccess(0)	(0x555d75b0fb80, {
43842	cuLaunch	rKernel			CUDA_SUCCESS(0)	(0x555d76ef97f0, 11
43843	device	_tea_leaf_ppcg_solve_init	device_tea_leaf_ppc	g_solve_init	cudaSuccess(0)	
43844	cudaLaunch	rKernel			cudaSuccess(0)	(0x555d75afca40, (
43845	cuLaunch	rKernel			CUDA_SUCCESS(0)	(0x555d76ec82a0, 9
43846	reduct	ion	reduction		cudaSuccess(0)	
43847	cudaLaunch	rKernel			cudaSuccess(0)	(0x555d75afca40, {{
43848	cuLaunch	rKernel			CUDA_SUCCESS(0)	(0x555d76ec82a0, 8
43849	reduct	ion	reduction		cudaSuccess(0)	
43850	cudaLaunch	rKernel			cudaSuccess(0)	(0x555d75afca40, {
43851	cuLaunch	nKernel			CUDA_SUCCESS(0)	(0x555d76ec82a0, 1
43852	reduct	ion	reduction		cudaSuccess(0)	
43853	cudaMemo	y v			cudaSuccess(0)	(0x7ffe2e710ce8, 0x
43854	cuMemc	yDtoH_v2			CUDA_SUCCESS(0)	(0x7ffe2e710ce8, 0x
43855	cudaLaunch	nKernel			cudaSuccess(0)	(0x555d75b0f140, {
43856	cuLaunch	rKernel			CUDA_SUCCESS(0)	(0x555d76f0b610, 1
43857	device,	_tea_leaf_ppcg_solve_init	device_tea_leaf_ppc	g_solve_init_sd_new	cudaSuccess(0)	
43858	cudaLaunch	rKernel				(0x555d75b0f020, {
43859	cuLaunch	rKernel				(0x555d76f06b50, 1 -
sources						
CUDA: Mei						
D ^ A	PI Call ID	Allocation type	Address	Size Requested		Device ID ^
			Total allocations: 41	Total size: 2.63 Gb	vtes	
	603	UNIFIED MEMORY ALL OC	0x7fec32000000	32	bytes 0x555d7653	50.60 0
	655	DEVICE MEMORY ALLOC	0x7fec2=000000	122 10 1	Invies 0x555d7653	
	666	DEVICE MEMORY ALLOC	0x7fcc22000000	122.10 1	Ibytes 0x555d7652	
	600	DEVICE MEMORY ALLOC	0x71e022000000	122.19 1	luden 0x55547653	
	6//	DEVICE MEMORY ALLOC	0x7lec1a000000	122.19 K	uytes 0x05507653	0
	688	DEVICE MEMORY ALLOC	0x/fec12000000	122.19 N	lbytes 0x555d7653	0
ections/R	ules Info	API Statistics NVTX	Resources CPU (Call Stack		

(Non-interactive) Profile

Activity

🗠 Profile

🗟 Occupant

Cancel Reset Activity

Command Line

Profile	Profile an application using the is not supported for this activity	command line profiler. All GPU workloads are serialized. Note: Attach
Calculator	Supported APIs: CUDA	
	Common Filter	Sections Sampling Other
	Output File:	report_%i
	Force Overwrite:	
	Target Processes:	Application Only 👻
	Replay Mode:	Kernel 👻
	Application Replay Match:	
	Application Replay Buffer:	
	Command Line:	/tmp/var/target/linux-desktop-glibc_2_11_3x64/ncu – export/tmp/var/teport_%i-force-overwrite-target- processes application-only-replay-mode kernel – kernel- name-base function –launch-skip-before-match 0 –launch-

\$~/working/git/TeaLeaf_CUDA\$ /tmp/var/target/linux-desktop-glibc_2_11_3-x64/ncu -c 2 -k "regex:device tea leaf ppcg solve_(calc|update).*" ./tea leaf ==PROF== Connected to process 15827 (/home/fschmitt/working/git/Tealeaf CUDA/tea leaf) Output file tea.out opened. All output will go there. CUDA in rank 0 using NVIDIA GeForce RTX 2080 Ti Solver to use: PPCG Preconditioner to use: None Step 1 time 0.0000000 timestep 4.00E-03 Switching after 990 CG its, error 0.9911110E+00 Eigen min 0.106142E+01 Eigen max 0.537593E+05 Condition number 50648.403204 Error 0.995546E+00 ==PROF== Profiling "device tea leaf ppcg solve update r": 0%....50%....100% - 8 passes ==PROF== Profiling "device tea leaf ppcg solve calc sd new": 0%....50%....100% - 8 passes [15827] tea_leaf@127.0.0.1 device tea leaf ppcg solve update r(kernel info t, double *, const double *, const double *, const double *), 2021-Dec-14 14:02:35, Context 1, Stream 7 Section: GPU Speed Of Light Throughput DRAM Frequency cvcle/nsecond SM Frequency cycle/nsecond Elapsed Cycles cycle 1,457,762 Memory [%] % DRAM Throughput 96 Duration msecond L1/TEX Cache Throughput L2 Cache Throughput 90

INF The kernel is utilizing greater than 80.0% of the available compute or memory performance of the device. To further improve performance, work will likely need to be shifted from the most utilized to another unit. Start by analyzing workloads in the Memory Workload Analysis section.

SM Active Cycles

Compute (SM) [%]

cycle

8

6.63

1.33

91.70

91.70

1.09

28.58

39,02

86.51

1,451,446.41

<u>P</u> age: Summary	- Launch	n: 0 - 43843 - device_tea_leaf	_ppcg_sol 👻	🛛 🝷 Add Baseline	 Apply <u>R</u>ules 	🗟 🗔 Occupancy O	alculator					Сору	as Image 👻
	Launch			Time	Cycles R	egs GPU		SM Fre	quenc	CC F	Process		
Current	43843 - de	vice_tea_leaf_ppcg_solve_init	(126, 1001, 1)x	(32, 4, 1) 217.63 used	ond 297,114 4	0 0 - NVIDIA Gel	orce RTX 2080 1	Fi 1.36 cyc	cle/nse	- cond 7.5 [15958] tea_leaf		
11													
ID 🔺 Time	API Call ID	Function Name	Demangled N	Process	Device Name	Grid Size	Block Si	ze		Cycles [cyc	le] Duration [msecond] (Compute Throughput	t [%] Memc
0 2021-Dec	4384	3 device_tea_leaf_ppcg_	device_te_	[15958] tea_leaf	NVIDIA GeFor	rce 126, 10	01, 1	32,	4,	1 297,1	14 0.22	7	7.89
1 2021-Dec-1	438	57 device_tea_leaf_ppcg_sol	device_tea_I	. [15958] tea_leaf	NVIDIA GeForce	RT 126, 1	001, 1	32	, 4,	1 1,264,9	0.94		54.78
2 2021-Dec-1	438	50 device_tea_leaf_ppcg_sol	device_tea_I	. [15958] tea_leaf	NVIDIA GeForce	RT 126, 1	001, 1	32	, 4,	1 1,462,4	46 1.07	1	86.22
3 2021-Dec-1	438	53 device_tea_leaf_ppcg_sol	device_tea_I	. [15958] tea_leaf	NVIDIA GeForce	RT 126, 1	001, 1	32	, 4,	1 1,443,8	336 1.06	:	23.81
1 1		'											
		· · · · · · · · · · · · · · · · · · ·											
<u>P</u> age: Details	▼ Launcl	n: 0 - 43843 - device_tea_leaf	_ppcg_sol [*] *	Add Baseline	Apply <u>R</u> ules	🗄 Occupancy C	alculator					Сору а	s Image 👻
	Launch			Time	Cycles R	egs GPU		SM Fre	quenc	Y CC P	rocess	\oplus Θ	0
Current	43843 - de	wice_tea_leaf_ppcg_solve_init	(126, 1001, 1)x	(32, 4, 1) 217.63 used	ond 297,114 4	0 0 - NVIDIA Gef	orce RTX 2080 T	ï 1.36 cyc	le/nsec	cond 7.5 [1	5958] tea_leaf		
GPU Speed Of L	ight Throug	jhput											ρ
Compute (SM) Throu	ghput [%]				77.89	Duration [usecon	d]					2	217.63
Memory Throughput	[%]				45.03	Elapsed Cycles [c	ycle]					29	97,114
L1/TEX Cache Throu	ghput [%]				68.22	2 SM Active Cycles	[cycle]					293,3	85.19
L2 Cache Throughpu	it [%]				2.30	SM Frequency [cy	cle/nsecond]						1.36
DRAM Throughput [%	6]				0.12	2 DRAM Frequency	[cycle/nsecond]						6.79
🔥 High Compu	ute Through	Dut Compute is more heav	ily utilized than	Memory: Look at the	Compute Workloa	<u>d Analysis</u> report se	tion to see what	the compu	ite pipe	elines are spe	nding their time doing. Also	, consider	\odot
		 wnetner any computat 	ion is redundar	nt and could be reduce	a or moved to loo	K-up tables.							
	т	bo ratio of poak float (fp22) to	double (fp64)	porformance on this d	ovico is 22:1 Tho	kornal achieved 0%	of this dovico's fo	22 pook po	rformo	unco and 10%	of its fn64 nook performan	co. If Compute	
🔥 FP64/32 Uti	ilization 🖞	orkload Analysis determines	that this kernel	is fp64 bound, consid	er using 32-bit pre	cision floating point	operations to in	oz peak pe prove its p	erform	ance. See the	Kernel Profiling Guide for n	node details on roofli	ine
	а	nalysis.											
- Compute Workl	oad Analys	is											
Executed Inc Flansed	[inst/cvcle]				1.25	5 SM Busy [%]							78.61
Executed lpc Active li	inst/cvclel				1.2	/ Issue Slots Busy	%]						31.65
Issued Ipc Active [ins	t/cycle]				1.27	7							

Selec	cted result		Metric va	lues		
Page: Details 🔹 Launch: 0 - 438	43 - device_tea_leaf_ppcg_sol 🔹 🍸 👻 Add Baseline 🔹 🖌	Apply <u>R</u> ules 🔲 Occupancy Calculator			Copy as Image 👻	
Launch Current 43843 - device tea lé	Time (Cycles Regs GPU 297.114_40 0 - NVIDIA GeForce RTX 2080 Ti	SM Frequency CC Prod	c ess 58) tea leaf	\odot \odot \odot	
GPU Speed Of Light Throughput				All		
Compute (SM) Throughput [%]		77.89 Duration [usecond]			217.63	
Memory Throughput [%]		45.03 Elapsed Cycles [cycle]			297,114	
L1/TEX Cache Throughput [%]		68.22 SM Active Cycles [cycle]			293,385.19	
L2 Cache I nroughput [%]		2.30 SM Frequency [cycle/nsecond]			6.70	
		0.12 Drawn requercy (cycle/haccond)			0.79	
▲ High Compute Throughput ^{Cor} wha	npute is more heavily utilized than Memory: Look at the <u>Compu</u> ther any computation is redundant and could be reduced or m	<u>ute Workload Analysis</u> report section to see what t oved to look-up tables.	he compute pipelines are spendi	ng their time doing. Also, consider	\odot	
▲ FP64/32 Utilization The ratio of Workload A analysis.	peak float (fp32) to double (fp64) performance on this device is nalyzis determines that this kernel is fp64 bound, consider usin —	s 32:1. The kernel achieved 0% of this device's fp3 g 32-bit precision floating point operations to imp	2 peak performance and 19% of prove its performance. See the <u>Ke</u>	its fp64 peak performance. If <u>Com</u> r <u>rr el Profiling Guide</u> for mode detai	<u>oute</u> ils on roofline	
🗕 Compute Workload Analysis	7				Q	
Executed Ipc Elapsed [Inst/cycle]		1.25 SM Busy [%]			78.61	
Executed Ipc Active [inst/cycle]		1.27 Issue Slots Busy [%]			31.65	
Issued Ipc Active [inst/cycle]		1.27				
Expandable			Expert	Analysis		
Sections			(Ru	ules)		



Detailed memory workload analysis chart and tables

Shows transferred data or throughputs

Tooltips provide metric names, calculation formulas and detailed background info

Current	655 - reduceFinal (1, 1, 1)x(5	512, 1, 1) 5.98 usecond	4,807 16	NVIDIA GeFo	orce RTX 2080 Ti 8	301.61 cycle/usecond	7.5 [4969] simpleC	udaGraphs		
Baseline	654 - reduce (512, 1, 1)x(51)	2, 1, 1) 234.72 usecor	nd 318,783 16	NVIDIA GeFo	orce RTX 2080 Ti 1	1.35 cycle/nsecond	7.5 [4969] simpleC	udaGraphs		
- GPU Speed Of Light	t Throughput							GPU Th	oughput Char	γρ
Compute (SM) Throug	hout [%]			0.33 (-99.58%)	Duration Jusecond	4]				5 98 (-97 45%)
Memory Throughput [9	%] %]			0.80 (-98.25%)	Elapsed Cycles [c	vclel				4.807 (-98.49%)
L1/TEX Cache Through	hput [%]		11	.66 (+12.76%)	SM Active Cycles	[cvcle]				46.09 (-99.98%)
L2 Cache Throughput	[%]			0.80 (-94.49%)	SM Frequency (cy	cle/usecond]			8	01.61 (-40.80%)
DRAM Throughput [%]				0.48 (-98.94%)	DRAM Frequency	[cycle/nsecond]				3.98 (-41.14%)
▲ Small Grld Ti	his kernel grid is too small to fi sis The ratio of peak float (fi	ill the available resources (p32) to double (fp64) perfo	on this device, resi ormance on this de	ulting in only 0.0) full waves across a	all SMs. Look at <u>Laun</u> % of this device's fp3:	<u>ch Statistics</u> for more 2 peak performance ar	details. nd close to 0% of it:	s fp64 peak pe	♥ Iformance.
				GPU TI	hroughput					
т										
Compute (SM) [%]										
			_	_						
Momony [%]					_					
wierriory [76]										
+ 0.0) 10.0	20.0	30.0	40.0 Sp	50.0 beed Of Light (SOL)	60.0 [%]	70.0	80.0	90.0	100.0
Compute Workload	Analysis									0
Frequited Inc Flanced	(inot (ovolo)			0.00 (00.16%)	CM Duoy [9]					24 72 (50 02%)
Executed Inc Active In	(IIISt/Cycle) het/cycle]			0.00 (-99.10%)	Sivi Busy [/6]	શ્રી				54.72 (-59.95%) 7.11 (-14.71%)
Issued Inc Active linst	/cvcle]			0.20 (-20.00%)	lissue olots busy [
 Balanced No 	pipeline is over-utilized.			0.20 (11110)						
- Memory Workload	Analysis								All	×Ω
Memory Throughout [0	Shyte/second]			1 84 (-99 38%)	Mem Busy [%]					0.80 (-94.49%)
L1/TEX Hit Rate [%]				0 (-100.00%)	Max Bandwidth [%	6]				0.57 (-98.75%)
L2 Hit Rate [%]			71.25	(+43,797.06%)	Mem Pipes Busy					0.11 (-99.15%)
				Memo	ory Chart					
			16.00 Reg							
-	17.00 Inst (100.00%)	(-100.00%)						⊣ ≝ 💼 100%	
	(*100.00%)	Global					0.00 B	<u>≻</u>	Pea	
			1.00 Req				(+0.00%)	e	*	
			(-99.80%)					→ ×		
	0.00 Inst		(+0.00%)				0.00 B	ster	80%	
	(+0.00%)	Local					(+0.00%)	Ś		
			0.00 Req (+0.00%)	L1/TEX Cache	4.00 KB (-99.99%)	12 Cacha				

Comparison of results directly within the tool with "Baselines"

Supported across kernels, reports, and GPU architectures

Source: cuda_fp16.hpp 🔹 🖻 Find			s	Source:half2		• 8	Find	× ^			
Navigation: Sampling Data (All) - · · · 조 답	년 전			Navigation: Sampling D	oata (All)			~ 주 18 년			
# Cource	Sampling Sampling Data	Instructions Prec		# Addross	Source	•		Sampling	Sampling Data	Instructions	Predicated-On Three
305 struct CUDA ALIGN (4) half2 {		Executed matt		40 00007f25 cef9e	d70	NOP		0	(Not issued)	CXecuted 0	Instructions Executi
306 half x:	0 0				half	í2					
307half y;				1 00007f25 cefa2	d00	MOV R7,		73	71	34,944	1,081,21
				2 00007f25 cefa2	d10	MOV R6,		37	30	34,944	1,081,21
309 // All construct/copy/assign/move				3 00007f25 cefa2	d20	MOV R5,		37	32	34,944	1,081,21
				4 00007f25 cefa2	d30	MOV R4,		33	31	34,944	1,081,21
<pre>311 #if defined(CPP_VERSION_AT_LEAST_11_FP16)</pre>				5 00007f25 cefa2	d40	MOV R4,		30	24	34,944	1,081,21
				6 00007f25 cefa2	d50	MOV R5,		40	37	34,944	1,081,21
313CUDA_HOSTDEVICEhalf2(consthalf2 &&src)	1,010 913	800,768		7 00007f25 cefa2	d60	MOV R8,		34	31	34,944	1,081,21
314CUDA_HOSTDEVICEhalf2 &operator=(constha	1,056 951	884,736		8 00007f25 cefa2	d70	MOV R9,		29	26	34,944	1,081,21
	0 0			9 00007f25 cefa2	d80	MOV R6,		27	26	34,944	1,081,21
316CUDA_HOSTDEVICEhalf2() { }				10 00007f25 cefa2	d90	MOV R7,		48	37	34,944	1,081,21
317 #endif /* defined(CPP_VERSION_AT_LEAST_11_FP16) */				11 00007f25 cefa2	da0	MOV R4,		42	35	34,944	1,081,21
318CUDA_HOSTDEVICEhalf2(consthalf &a, con	6 0 58	19,456		12 00007f25 cefa2	db0	MOV R5,		49	44	34,944	1,081,21
<pre>319CUDA_HOSTDEVICEhalf2(consthalf2 &src) {</pre>	1,371 1,287	698,880		13 00007f25 cefa2	dc0	MOV R4,		39	36	34,944	1,081,21
320CUDA_HOSTDEVICEhalf2 &operator=(constha	Total Sample Count: 137	72,704		14 00007f25 cefa2	dd0	MOV R5,		44	44	34,944	1,081,21
	3.14% Selected (43)	4)		15 00007f25 cefa2	de0	LD.E.SYS		36	3	34,944	1,081,21
	6.49% Branch Resolving	(89)		16 00007f25 cefa2	df0	MOV R6,		531	521	34,944	1,081,21
323CUDA_HOSTDEVICEhalf2(consthalf2_raw &h:	35.81% Long Scoreboard 51.35% Wait (704)	(491)		17 00007f25 cefa2	e00	MOV R7,		38	36	34,944	1,081,21
324CUDA_HOSTDEVICEhalf2 &operator=(constha	0 0			18 00007f25 cefa2	e10	ST.E.SYS		35	38	34,944	1,081,21
325CUDA_HOSTDEVICE operatorhalf2_raw() const				19 00007f25 cefa2	e20			57	50	34,944	1,081,21
				20 00007f25 cefa2	e30	RET.ABS.	NODEC R20 0x0	34	31	34,944	1,081,21
				21 00007f25 cefa2	e40			83	78		
				22 00007f25 cefa2	e50	NOP					
329 #if defined(CUDACC)				23 00007f25 cefa2	e60	NOP					
				24 00007f25 cefa2	e70	NOP					
					_half	f2float					
332 #if (!defined(CUDA_ARCH) (CUDA_ARCH \geqslant 536				1 00007f25 cefb6	f00	IADD3 R1		18	18	256	25
				2 00007f25 cefb6	f10	S2R R0,		4	4	256	25
334deviceforceinlinehalf2 operator+(const	709 631	534,528		3 00007f25 cefb6	F20	ISETP.GE	.U32.AND P0, P1			256	25
335deviceforceinlinehalf2 operator-(const				4 00007f25 cefb6	f30 @P0			4	4	256	25
336deviceforceinlinehalf2 operator*(const	391 343	679,936		5 00007f25 cefb6	f40	BPT.TRAP					

Source/PTX/SASS analysis and correlation

Source metrics per instruction and aggregated (e.g. PC sampling data)

Metric heatmap

Occupancy Calculator

Compute Capability: Shared Memory Size Config (bytes): CUDA version: Global Load Cache Mode:	7.5 65536 11.0 L1+L2 (ca)	 Threa Regist Share 	ds Per Block: ers Per Thread: d Memory Per Blo	12 40 ock (bytes): 30	28 • • • •		
Tables Graphs Gl	PU Data			Арр	ly Reset		
Occupancy Data:						Physical Limit of GPU (7.5):	
Property			Value			Property	Limit
Active Threads per Multiprocessor					102	Threads per Warp	32
Active Warps per Multiprocessor					3:	Max Warps per Multiprocessor	32
Active Thread Blocks per Multiproce	essor				:	Max Thread Blocks per Multiprocessor	16
Occupancy of each Multiprocessor					100 %	Max Threads per Multiprocessor	1024
						Maximum Thread Block Size	1024
Allocated Resources:						Registers per Multiprocessor	65536
Resources		Per Block	Limit Per SM	Allocatab	le Blocks Per SM	Max Registers per Thread Block	65536
Warps (Threads Per Block / Thread	ls Per Warp)		4 32		:	Max Registers per Thread	255
Registers (Warp limit per SM due to	per-warp reg count		4 48			Shared Memory per Multiprocessor (bytes)	65536
Shared Memory (Bytes)		307	2 65536		2	Max Shared Memory per Block	65536
						Register Allocation Unit Size	256
						Register Allocation Granularity	warp
Occupancy Limiters:						Shared Memory Allocation Unit Size	256
Limited By	Blocks	oer SM _ W	arps Per Block	War	ps Per SM	Warp Allocation Granularity	
Max Warps or Max Blocks per Multi	iprocessor			Δ	3:	Shared Memory Per Block (bytes) (CUDA runtime use)	
Registers per Multiprocessor				The occup	ancy is limited by b	ock size	
Shared Memory per Multiprocessor							

Hands-On with TeaLeaf_Cuda

Get the application

Get the source code

\$ git clone --depth 1 https://github.com/UK-MAC/TeaLeaf CUDA

Get compiler, MPI and CUDA using modules

module load Stages/2022
module load NVHPC/22.1
module load OpenMPI/4.1.2
module load Nsight-Compute/2022.1.0

Update Makefile for the target architecture (e.g. AMPERE, SM 8.0) and compiler/libraries, as necessary

Build the application

\$ make

Better, use the pre-compiled script and binary from /p/project/training2123/work/schmitt5 \$ source load_modules.sh

\$ TeaLeaf_CUDA/tea_leaf

Overview with Nsight Systems

Profile with Nsight Systems to identify best CUDA kernel optimization targets Focus on device_tea_leaf_ppcg_solve_(calc|update).* kernels



Profiling with Nsight Compute

Collect 10 instances of those kernels with ncu, full set of metrics Inspect the resulting report in the Nsight Compute UI (ncu-ui)

ncu --set full -k "regex:device_tea_leaf_ppcg_solve_(calc|update).*" -c 10 -f -o tea_leaf_%i
./tea_leaf

Summary page confirms that all instances of each respective kernel have similar performance characteristics - focus on a single instance for each

<u>P</u> age	Summary	<u>R</u> esult: 0 - 32940 - devic	ce_tea_leaf_ppcg_so	ol 👻 🍸 👻 Add Ba	seline 👻 Apply	<u>R</u> ules	🖬 Occ	upancy Calculato	r			Copy as Image
		Result			Time	Cycles	Regs	GPU	SM F	requency CC	Process	
	Current	32940 - device_tea_leaf_ppc	g_solve_update_r (1)	26, 1001, 1)x(32, 4, 1)	475.90 usecond	512,011	32	0 - NVIDIA A10	-SXM4-40GB 1.08	cycle/nsecond 8.0	[9073] tea_leaf	
tą	 Use the column 	headers to sort the results in	this report. Double-c	click a result to see det	ailed metrics.							
	 Function Name 		Demangled Na Pro	ocess Device Name	Grid Size	Block Si	ze	Cycles [cycle]	Duration [usecond]	Compute Throughp	out [%] Memory	Throughput [%]
	0 device_tea_leaf	f_ppcg_solve_update_r	device_tea_I [90	073] t NVIDIA A100	126, 1001,	1 32,	4, 1	512,011	475.90		12.94	87.08
	1 device_tea_leaf	_ppcg_solve_calc_sd_new	device_tea_l [90	073] t NVIDIA A100	126, 1001,	1 32,	4, 1	548,244	502.24		11.05	79.95
	2 device_tea_leaf	_ppcg_solve_update_r	device_tea_l [90	073] t NVIDIA A100	126, 1001,	1 32,	4, 1	512,359	474.69		12.93	87.03
	3 device_tea_leaf	_ppcg_solve_calc_sd_new	device_tea_I [90	073] t NVIDIA A100	126, 1001,	1 32,	4, 1	546,152	500.74		11.10	80.26
	4 device_tea_leaf	f_ppcg_solve_update_r	device_tea_I [90	073] t NVIDIA A100	126, 1001,	1 32,	4, 1	511,999	470.53		12.94	87.09
	5 device_tea_leaf	f_ppcg_solve_calc_sd_new	device_tea_I [90	073] t NVIDIA A100		1 32,	4, 1	545,719	500.48		11.10	80.33
	6 device_tea_leaf	f_ppcg_solve_update_r	device_tea_l [90	073] t NVIDIA A100	. 126, 1001,	1 32,	4, 1	513,311	471.68		12.90	86.86
	7 device_tea_leaf	_ppcg_solve_calc_sd_new	device_tea_l [90	073] t NVIDIA A100		1 32,	4, 1	548,179	498.98		11.06	79.97
	8 device_tea_leaf	_ppcg_solve_update_r	device_tea_I [90	073] t NVIDIA A100	126, 1001,	1 32,	4, 1	513,820	469.86		12.89	86.78
	9 device_tea_leaf	_ppcg_solve_calc_sd_new	device_tea_I [90	073] t NVIDIA A100	126, 1001,	1 32,	4, 1	547,310	503.71		11.07	80.09

Α

Switch to the *Details* page and select the second kernel (with slightly worse throughputs). Memory units are over-utilized.

Roofline shows that floating point performance is memory-bound (left of ridge point) Similar in Compute Workload chart: FP pipelines are less than 5% utilized.



MWA table shows bandwidth 80% utilized, chart shows high Device-to-L2 utilization



Scheduler stats show low eligible/issued, need to check stall reasons

Good achieved occupancy, doesn't appear to be the issue

Stall reasons dominated by long scoreboard, locate using Source Counters section



MWA found sub-optimal cache access patterns, locate using Source Counters section

Source Counters show uncoalesced accesses and location of the stalls

Jump to Source page via this link

Δ	L1TEX Global Store Acces	s Pattern	The memory access pattern f pattern, possibly caused by th thread address pattern for 8.0 the <u>Source Counters</u> section f	or global stores in LTTEX might not be optimal. On average, this kernel accesses 8.0 bytes per thread per memory request, but the addres: e stride between threads, results in 9.0 sectors per request, or 9.0°32 = 288.0 bytes of cache data transfers per request. The optimal byte accesses would result in 8.0°32 = 256.0 bytes of cache data transfers per request, to maximize L1TEX cache performance.Check for uncoalesced global stores.	s
•	L2 Store Access Pattern	The mem per L2 rec stores an	ory access pattern for stores fr juest. However, this kernel only d try to minimize how many cao	om L1TEX to L2 is not optimal. The granularity of an L1TEX request to L2 is a 128 byte cache line. That is 4 consecutive 32-byte sectors accesses an average of 3.0 sectors out of the possible 4 sectors per cache line. Check the <u>Source Counters</u> section for uncoalesced che lines need to be accessed per memory request.	Θ
•	L2 Load Access Pattern	The memo L2 reques and try to	ory access pattern for loads fro t. However, this kernel only acc minimize how many cache line	m L1TEX to L2 is not optimal. The granularity of an L1TEX request to L2 is a 128 byte cache line. That is 4 consecutive 32-byte sectors per esses an average of 3.0 sectors out of the possible 4 sectors per cache line. Check the <u>Source Counters</u> section for uncoalesced loads s need to be accessed per memory request.	r O
▼ Sc	ource Counters			All	- p
Bran	ch Instructions [inst]			2,518,016 Branch Efficiency [%]	100
Bran	ch Instructions Ratio [%]			0.10 Avg. Divergent Branches	
•	Uncoalesced Global Acce	sses Exc	s kernel has uncoalesced globa essive table for the primary sou	I accesses resulting in a total of 2500000 excessive sectors (11% of the total 2500000 sectors). Check the L2 Theoretical Sectors Global irce locations. The <u>GUDA Programming Guide</u> had additional information on reducing uncoalesced device memory accesses. L2 Theoretical Sectors Global Excessive	Θ
Loca	tion			Value	Value (%)
<u>0x14</u>			ve_calc_sd_new *	500,000	20
<u>0x14</u>			ve_calc_sd_new	500,000	20
0x14			ve_calc_sd_new A	500,000	20
0x14			Ive_calc_sd_new		20
0.14			ve_calc_su_new *	Warp Stall Sampling (All Cycles)	20)
Lo	cation			Value	Value (%)
0x	14ff32fbfdc0 in device_tea_l			26.623	47
0x				26,144	46
0x				1,307	2
<u>0x</u>				664	1
<u>0x</u>				332	1

Stalled at DMUL instruction, waiting for LDG (load global) in line 43 (via register R10) LDG instructions are uncoalesced Lots of excessive (non-ideal) L2 sector accesses

View	SASS 🔻															
Source	e: device_tea	_leaf_ppcg_s	olve_calc_	_sd_new 💌 🖃 Find	Navigati	ion: Warp St	all Sampling (All Cy	eles) 👻	라마수~	' ይ			es			
	‡ ∆ddress		Source				Live Registers	Warp Stall Sampling (All Cycles)	Warp Stall Sampling (Not-issued Cycles)	Instructions Executed	L2 Theoretical Sectors Global Excessive	L2 Theoretical			Re	gister
30	000014ff	32fbfcd0	1	ISETP.GT.OR PO, PT,	R3. UR5. P0		3	56	24	504.504					Dopondo	
31	000014ff	32fbfce0	@P0 8				a	135	55	504,504						
	000014ff	32fbfcf0		JLDC UR4, c[0x0][0x1	64]			48	20	500,000						
	000014ff	32fbfd00	٩	10V R2, c[0x0][0x1f0			Ĭ	16		500,000						
	000014ff	32fbfd10		JLEA UR4, UR7, UR4,			<u> </u>			500,000						
	000014ff	32fbfd20	1	IMAD.MOV.U32 R15, RZ			<u> </u>			500,000						
	000014ff	32fbfd30	1	IMAD.WIDE R8, R15, R	2, c[0x0][0x1e8]			42		500,000						
	000014ff	32fbfd40	1	IMAD R0, R4, UR4, R3				39	14	500,000						
	000014ff	32fbfd50	1	IMAD.WIDE R2, R15, R	2, c[0x0][0x1e0]			42		500,000						
	000014ff	32fbfd60		LDG.E.64.CONSTANT R8	, [R8. <mark>64</mark>]			75	60	500,000		500,000				
	000014ff	32fbfd70	1	IMAD.WIDE R10, R0, R	15, c[0x0][0x1a8]		9	25		500,000						
	000014ff	32fbfd80		LDG.E.64.CONSTANT R2	, [R2. <mark>64</mark>]		9	56	26	500,000		500,000				
	000014ff	32fbfd90	1	IMAD.WIDE R4, R0, R1	5, c[0x0][0x198]		11	28		500,000						
	000014ff	32fbfda0		LDG.E.64.CONSTANT R1	0, [R10 <mark>.64</mark>]		11	64	24	500,000	500,000	4,500,000			•	
	000014ff	32fbfdb0		LDG.E.64 R6, [R4.64]			13		48	500,000	500,000	4,500,000			│ Þ¶ ¶	
45	5 000014ff	32fbfdc0		DMUL R12, R10, R8			15	26,623	23,471	500,000				þ þ þ		1
	000014ff	32fbfdd0	0	DFMA R6, R6, R2, R12			11	664	598	500,000					 	
	000014ff	32fbfde0	I	IMAD.WIDE R12, R0, R	15, c[0x0][0x1b0]		9			500,000						
	000014ff	32fbfdf0		STG.E.64 [R4.64], R6			_7	75	60	500,000	500,000	4,500,000				
	000014ff	32fbfe00		LDG.E.64 R14, [R12.6				43		500,000	500,000	4,500,000				
50	000014ff	32fbfe10	C	DADD R14, R6, R14			7	26,144	23,172	500,000					ÞÞ	

Where is this in the code (no CUDA-C source) correlation?

Need to add *-lineinfo* flag in Makefile during compilation on line 145 (NV_FLAGS), re-compile, re-run

Consider using --import-source yes

_						
\	ew: Source and SASS 👻					
So	urce: tea_leaf_ppcg.cuknl 💽 🖻 Find		s	Source: device_tea_leaf_ppcg_	.solve_calc_sd_new 🔽 😑 Find	
Na	vigation: Warp Stall Sampling (All Cycles) 🔹 🗸 추 담 담 오		١	Navigation: Warp Stall Sampli	ing (All Cycles) 주 旧 묘 오	
	# Source	Warp Stall Sampling a ^ (All Cycles) 4		# Address	Source	Warp Stall Sampling
	231 + beta[step]*rtemp[THARR2D(37 00001476 e6fc4f40	IMAD RO, R4, UR4, R3	44
	232 utemp[THARR2D(0, 0, 0)] += sd[THARR2D(0, 0, 0			38 00001476 e6fc4f50	IMAD.WIDE R2, R15, R2, c[0x0][0x1e0]	36
				39 00001476 e6fc4f60	D LDG.E.64.CONSTANT R8, [R8.64]	81
				40 00001476 e6fc4f70	IMAD.WIDE R10, R0, R15, c[0x0][0x1a8]	
	<pre>235 else if (PRECONDITIONER = TL_PREC_NONE)</pre>			· 41 00001476 e6fc4f80	D LDG.E.64.CONSTANT R2, [R2.64]	
				42 00001476 e6fc4f90	<pre>IMAD.WIDE R4, R0, R15, c[0x0][0x198]</pre>	
	237 if (WITHIN_BOUNDS)	492		43 00001476 e6fc4fa0	D LDG.E.64.CONSTANT R10, [R10.64]	66
				· 44 00001476 e6fc4fb0	D LDG.E.64 R6, [R4.64]	
•	239 sd[THARR2D(0, 0, 0)] = alpha[step]*sd[THARR2D	27,842		> 45 00001476 e6fc4fc0	DMUL R12, R10, R8	26,586
	240 + beta[step]*rtemp[THARR2D(46 00001476 e6fc4fd0	DFMA R6, R6, R2, R12	650
	241 utemp[THARR2D(0, 0, 0)] += sd[THARR2D(0, 0, 0	26,351		47 00001476 e6fc4fe0	IMAD.WIDE R12, R0, R15, c[0x0][0x1b0]	20
				48 00001476 e6fc4ff0	STG.E.64 [R4.64], R6	
				49 00001476 e6fc5000	D LDG.E.64 R14, [R12.64]	40
				50 00001476 e6fc5010	DADD R14, R6, R14	26,176
				51 00001476 e6fc5020	STG.E.64 [R12.64], R14	107
				52 00001476 e6fc5030		8
	247 /* New ppcg_store_r */			53 00001476 e6fc5040	ULDC UR7, c[0x0][0x170]	1,409
				54 00001476 e6fc5050	ULDC UR4, c[0x0][0x16c]	0
	249global void device_tea_leaf_ppcg_store_r			55 00001476 e6fc5060	UIADD3 UR4, UR7, UR4, URZ	0
	250 (kernel_info_t kernel_info,	0 💌		56 00001476 e6fc5070	ULDC UR6, c[0x0][0x18c]	0 💌
_						

Further CLI exercises

Check collected report on command line using ncu -i and --page for comparison with the UI.

Update tea_leaf_kernel_cuda.cu with a nvtxPush/Pop range around the two kernels Name the range "update_and_calc" Include <nvtx3/nvToolsExt.h> Update Makefile with "-ldl" at line 134 Replace -k "..." ncu command line with "--nvtx --nvtx-include "update_and_calc/" and "--set full" with "--metrics ..."

More: Data Collection

Collecting Data

By default, CLI results are printed to stdout Use --export/-o to save results to a report file, use -f to force overwrite \$ ncu -f -o \$HOME/my_report <app> \$ my_report.ncu-rep

Use --log-file to pipe text output to a different stream (stdout/stderr/file)

Can use (env) variables available in your batch script or file macros to add report name placeholders Full parity with nvprof filename placeholders/file macros \$ ncu -f -o \$HOME/my_report_%h_\${LSB_JOBID}_%p <app> \$ my_report_host01_951697_123.ncu-rep

https://docs.nvidia.com/nsight-compute/NsightComputeCli/index.html#command-line-options-filemacros

What To Collect

Curated "sets" and "sections" with commonly-used, high-value metrics

\$ ncu --list-sets
Identifier Sections

Estimated Metrics

default	LaunchStats, Occupancy, SpeedOfLight	35
detailed	ComputeWorkloadAnalysis, InstructionStats, LaunchStats, MemoryWorkloadAnaly sis, Occupancy, SchedulerStats, SourceCounters, SpeedOfLight, SpeedOfLight_	157
full	ComputeWorkloadAnalysis, InstructionStats, LaunchStats, MemoryWorkloadAnaly sis, MemoryWorkloadAnalysis Chart, MemoryWorkloadAnalysis Tables, Occupancy	162
	, SchedulerStats, SourceCounters, SpeedOfLight, SpeedOfLight_RooflineChart, WarpStateStats	
		4 7

source SourceCounters

47

Use defaults, or combine as desired

\$ ncu --set default --section SourceCounters --metrics sm__inst_executed_pipe_tensor.sum ./my-app

What To Collect

Query metrics for any targeted chip

```
$ ncu --query-metrics --chip ga100
smsp warps issue stalled not selected
                                                                            cumulative # of warps waiting
for the microscheduler to select the warp to issue
smsp warps issue stalled selected
                                                                            cumulative # of warps selected
by the microscheduler to issue an instruction
smsp warps issue stalled short scoreboard
                                                                            cumulative # of warps waiting
for a scoreboard dependency on MIO operation other than (local, global, surface, tex)
tpc cycles active
                                                                            # of cycles where TPC was active
tpc cycles elapsed
                                                                            # of cycles where TPC was active
==PROF== Note that these metrics must be appended with a valid suffix before profiling them. See --help for
more information on --query-metrics-mode.
```

Specify sub-metrics in section files, or on the command line

```
$ ncu --query-metrics-mode suffix --metrics sm__inst_executed_pipe_tensor ./my-app
sm__inst_executed_pipe_tensor.sum
sm__inst_executed_pipe_tensor.avg
sm__inst_executed_pipe_tensor.min
```

Source Analysis

SASS (assembly) is always available, embedded into the report CUDA-C (Source) and PTX availability depends on compilation flags Use -lineinfo to include source/SASS correlation data in the binary

Source is not embedded in the report by default, need local or remote access to the source file to resolve in the UI. Import source during collection to (--import-source yes) to solve this.

Compiler optimizations can prevent exact source/SASS correlation

More: Data Analysis

Python Interfaces



Command Line Output

\$ ncu -i /tmp/report.ncu-rep -c 2 --page raw --csv --metrics \
gpc__cycles_elapsed.sum,sm__maximum_warps_per_active_cycle_pct

"ID","Process ID","Process Name","Host Name","Kernel Name","Kernel Time","Context","Stream","sm__maximum_warps_per_active_cycle_pct" "","","","","","","","","","%" "0","16301","tea_leaf","127.0.0.1","device_tea_leaf_ppcg_solve_update_r(kernel_info_t, double *, const double *, const double *, const double *)","2021-Dec-14 14:37:22","1","7","100.000000" "1","16301","tea_leaf","127.0.0.1","device_tea_leaf_ppcg_solve_calc_sd_new(kernel_info_t, const double *, double *, const double *, const double *, double *, const dou



On a single-node submission, one Nsight Compute instance can profile all launched child processes

Data for all processes is stored in one report file

ncu --target-processes all -o <singlereport-name> <app> <args>



On multi-node submissions, one tool instance can be used per node

Ensure that instances don't write to the same report file on a shared disk

ncu -o report_%q{OMPI_COMM_WORLD_RANK}
<app> <args>



Multiple tool instances on the same node are supported, but...

All kernels across all GPUs will be serialized using system-wide file lock

fi



Consider profiling only a single rank, e.g. using a wrapper script

```
#!/bin/bash
if [[ "$OMPI_COMM_WORLD_RANK" == "3" ]] ; then
    /sw/cluster/cuda/11.1/ nsight-compute/ncu -
o report_${OMPI_COMM_WORLD_RANK} --target-
processes all $*
else
    $*
```

34 🛛 🚳 NVIDIA

Conclusion

Conclusion

Nsight Compute enables detailed CUDA kernel analysis

Rules give guidance on optimization opportunities and help metric understanding

Limit metrics to what is required when overhead is a concern. Consider using application replay.

Still requires level of hardware understanding to fully utilize the tool - pay attention to rule results and refer to https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html

Analyze results in the UI, or post-process with CSV output or python report interface

Check known issues: <u>https://docs.nvidia.com/nsight-compute/ReleaseNotes/index.html#known-issues</u>

Further Reading

Download	https://developer.nvidia.com/nsight-compute (can be newer than toolkit version)
Documentation	<u>https://docs.nvidia.com/nsight-compute</u> (and local with the tool) <u>https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html</u>
Forums	https://devtalk.nvidia.com
Further Reading	https://developer.nvidia.com/nsight-compute-videos https://developer.nvidia.com/nsight-compute-blogs https://github.com/NVIDIA/nsight-training
	Dependency with integrative training metavial for multiple Maight tools, including

Repository with interactive training material for multiple Nsight tools, including Systems and Compute.

https://gitlab.com/NERSC/roofline-on-nvidia-gpus

