

## Nsight Compute OLCF Webinar

Felix Schmitt, Mahen Doshi, Jonathan Vincent

## **Nsight Product Family**

#### **Workflow**

Nsight Systems - Analyze application algorithms system-wide <u>https://www.olcf.ornl.gov/calendar/nvidia-profiling-tools-nsight-systems/</u>

Nsight Compute - Analyze CUDA kernels

Nsight Graphics - Debug/analyze graphics workloads



and the second se									22		
NVIDIA Nsight Compute										2 JL	, , ,
Connection <u>D</u> ebug <u>P</u> r O Connect (© Disconnect	rofile <u>T</u> ools <u>W</u> indow × Terminate [] 👲	Help Profile Kernel	* 🔊 💁 🛛 🕨								
old_2_fusion_on_softmax.r	nsight-cuprof-report * >										
age: Details 🐨 Proce	ecc. ΔI	▼ Launch: 0 - f	4791 - softmax_comp	te kernel 💌	Add Baseline * Ann	v Rules				Copy as	Image 🔻
		Lance Times				10 1/ CD CD Farm			(o.c.)		
Current 64291-so	ortmax_compute_kernel	(1966 Time:	15.65 usecond Cycle	s: 16,235 Regs:	28 GPU: Tesla V100-SXM	12-1668 Shirred	luency: 1.04 cyde/n	second CC: 7.0 PR	cess: [944] python	3.5 🕁	
▼ GPU Speed Of Light	A								SOL Chart		0
High-level overview of the u	itilization for compute an	d memory resource	es of the GPUL For each	unit the Speed O	flight (SOL) reports the ac	hieved percentage	of utilization with res	nect to the theoretical	maximum		
	raizadon for compare a	id meniory resource		and, the opeed of		aneveu percentage	of duizadoit widit ca	peer to the theoretical	ind Aintoin,		
SOL Memory [%]					2 42 Elanced Cycles	cyclel				16	225
SOL TEX [%]				5	5.37 SM Active Cycles	s [cvcle]				12.11	0.30
SOL L2 [%]				1	3.66 SM Frequency [cy	vcle/nsecond]					1.04
SOL FB [%]				4	3.42 Memory Frequency	[cycle/usecond	1]			78	1.94
				G	PII Utilization						
SM [%]											
Memory [%]											
0.0	10.0	20.0	30.0	40.0	50.0 Speed Of Light [%]	60.0	70.0	80.0	90.0	1	100.0
				Rec	ommendations						
A Bottleneck	[Warning] This kernel ex 50.0% of peak typically	hibits low compute indicate latency iss	throughput and memor ues. Look at `Schedule	y bandwidth utilizat r Statistics` and `V	ion relative to the peak pe Varp State Statistics`for p	rformance of this d otential reasons.	evice. Achieved comp	ute throughput and/or	memory bandwidth l	below	
Compute Workload A	Analysis										0
Detailed analysis of the comp performance.	pute resources of the st	reaming multiproce	ssors (SM), including th	e achieved instruct	ions per clock (IPC) and th	e utilization of each	available pipeline. Pip	belines with very high (	utilization might limit t	he overal	
Executed Ipc Elapsed [	inst/cycle]				1.83 SM Busy [%]						1.39
Executed Ipc Active [i	inst/cycle]				2.44 Issue Slots Busy	/ [%]					1.39
Issued Ipc Active [ins	t/cycle]				2.46 -						

CUDA Kernel profiler

Targeted metric sections for various performance aspects

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

UI and Command Line

Python-based rules for guided analysis (or post-processing)



#### Detailed memory workload analysis chart and tables



Comparison of results directly within the tool with "Baselines"

Supported across kernels, reports, and GPU architectures

oftma	xinlh T E & &		ZN5mxn	et2op8mxnet_op22softm	ax compute kernelīLi7ENS1 11softr	max fwdELb0EfLi2EN7msha	dow4half6half_tES6_EEvPT4_PT5
nstru	tions Executed				Sampling Data (Not Issued)		
		Sampling Data (AII) 🔺	= #	Source		Sampling Data (All)	Instructions Executed
Ξ,	# Judice	Sampling Data (All)	133	BSYNC RØ		Sumpling Data (Ail)	6.144
,	R5 global void softmax compute kernel/DTvne *in OTvne *out index t M	e	134	NOP		đ	6,144
2	Shapecodime Shapecodime	ě	135	BAR.SYNC 0x0		1	6,144
2	const double temperature) {	9	136	ISETP.GT.AND P		22	6,144
2	<pre>const unsigned x size = 1 &lt;&lt; x bits:</pre>	e _	137	BSSY B1, 0x7f8		1	6,144
2	39 shared AType smemfx sizel:	9	138	ISETP.GT.AND P	1, PT, R11, <b>0x1f</b> , PT	1	6,144
2	<pre>index t sa = stride[axis];</pre>	61	139	ISETP.GT.AND P	2, PT, R11, 0xf, PT	0	6,144
2	<pre>index t base = unravel dot(blockIdx.x, sshape, stride);</pre>	8	140	ISETP.GT.AND P	3, PT, R11, 0x7, PT	2	6,144
2	<pre>index t x = threadIdx.x;</pre>	52	141	@!P0 LDS.U R4, [R14-	+8x100]	2	6,144
2	13	8	142	@!P0 LDS.U R5, [R14		4	6,144
2	<pre>H4 red::maximum::SetInitValue(smem[x]);</pre>	44	143			3	6,144
2	<pre>if for (index_t i = x; i &lt; M; i += x_size) {</pre>	4	144	@!PO FMNMX R5, R5, H		2	6,144
2	<pre>smem[x] = ::max(smem[x], negate ? -in[base + i*sa] : in[base + i*sa</pre>	á	145			4	6,144
2	F7 }	0	146	NOP		0	6,144
2	<pre>#8syncthreads();</pre>	71	147	BAR.SYNC 0x0		4	6,144
	<pre>49 cuda::Reduce1D<red::maximum, x_bits="">(smem);</red::maximum,></pre>	111	148		+0x80]	18	6,144
2	<pre>50syncthreads();</pre>	Total Sample Cou	unt: 111	ISETP.GT.AND P		0	6,144
2	<pre>DType smax = smem[0];</pre>	Barrier: 43 (38.7%	5)	P1 LDS.U R7, [R14]		1	6,144
2	52syncthreads();	Mio Throttle: 21	(18.9%)	P1 STL [R1+0xc],		4	6,144
2		Not Selected: 8 (	7.2%)	P1 FMNMX R7, R7, I		0	6,144
2	<pre>54 red::sum::SetInitValue(smem[x]);</pre>	Selected: 7 ( 6.3%	6)	P1 STS [R14], R7		3	6,144
2	55 DType val;	Short Scoreboard	d: 16 (14.4)	%) NOP		4	6,144
2	56	Walt: 16 (14.4%)		BAR.SYNC 0X0		0	6,144
2	<pre>val = negate ? -in[base + i*sa]:in[base + i*sa];</pre>	10	156	@!P2 LDS.U R8, [R14	+0x40]	1	6,144
2	<pre>58 smem[x] += static_cast<atype>(expf((val - smax) / static_cast<atype< pre=""></atype<></atype></pre>	118	157	ISETP.GT.AND P	1, PT, R11, 0x1, PT	0	6,144
2	i9 }	0	158	@!P2 LDS.U R9, [R14		2	6,144
2	<pre>syncthreads();</pre>	4	159	eipz Sil [Ri+exie],	R8	4	6,144
2	<pre>51 cuda::Reduce1D<red::sum, x_d1ts="">(smem);</red::sum,></pre>	208	160	QUP2 FMNMX R9, R9, 1		0	6,144
2	<pre>synctnreads();</pre>	ᅨ	161	Eib5 [K14], K9		4	6,144
2 2	Alype Ssum = smem[0];	ě	162	Alpo Loc II 810 - E81	4+0v20]	1	6,144
- 4		1	160	Alphing upg upg [p14]		-1 -1	6 144
,	55 for (index t i = x; i < M; i += x size) {	-	164	@IP3 STI [81+0×14]		3	6.144
, <sup>2</sup>	$\frac{1}{2} = \frac{1}{2} + \frac{1}$	14	165	@IP3 EMNMX R5. R5.	R10. IPT	2	6,144
,	<pre>58 out[hase + i*sa] = OD::Map((va] = smax)/static cast/DTvpes(temperat</pre>		167	@[P3_STS_[R14], R5		2	6,144
2	<pre>59 }</pre>	ě –	168	NOP		- 0	6,144
		¥					

#### Source/PTX/SASS analysis and correlation

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

Metric heatmap

=PROF== Disconnected from proce [8792] CuBlackscholes.exe@127.0 GPUBlackScholesCallPut(int, f Section: GPU Speed Of Light	ess 8792 .0.1 loat*, float*	*, float*, floa	at*, float*), E	Block Size 256,
Metric Name	Metric Unit	Minimum	Maximum	Average
dramfrequency	Ghz	6.004059	6.004059	6.004059
fbpasol_pct	%	70.191350	70.191350	70.191350
<pre>gpcelapsed_cycles_max</pre>	cycle	751140.000000	751140.000000	751140.000000
<pre>gpcfrequency</pre>	Ghz	1.287365	1.287365	1.287365
<pre>gpucompute_memory_sol_pct</pre>	%	70.191350	70.191350	70.191350
<pre>gputime_duration</pre>	usecond	583.456000	583.456000	583.456000
ltcsol_pct	%	24.488190	24.488190	24.488190
<pre>smelapsed_cycles_avg</pre>	cycle	751121.000000	751121.000000	751121.000000
smsol_pct	%	69.036830	69.036830	69.036830
texsol_pct	%	20.449435	20.449435	20.449435

GPUBlackScholesCallPut(int, float*, float*, float*, float*, f Section: GPU Speed Of Light	loat*), 2019-Aug-12 14:44:50, Context 1	, Stream 7
Memory Frequency	Ghz	6.
SOL FB	%	71.
Elapsed Cycles	cycle	749,6
SM Frequency	Ghz	1.
Memory [%]	%	71.
Duration	usecond	580.
SOL L2	%	24.
SM Active Cycles	cycle	749,6
SM [%]	~ %	69.
SOLTEX	%	20.
OK Compute and Memory are well-balanced: To reduce runti Check both the `Compute Workload Analysis` and `Memor	me, both computation and memory traffic y Workload Analysis` report sections.	must be reduced.

Full command line interface (CLI) for data collection and analysis

#### On your workstation

Support for remote profiling across machines, platforms (Linux, Windows, ...) in UI and CLI

# Nsight Compute on Summit

### Loading Module

#### Use nv-nsight-cu-cli command line interface for data collection in batch environments

#### Available as part of the CUDA toolkit

\$ module load cuda/10.1.243

\$ /sw/summit/cuda/10.1.243/nsight-compute/nv-nsight-cu-cli

#### Or as standalone installation (e.g. newer release than CUDA)

\$ module load nsight-compute/2019.5.0

\$ /sw/summit/nsight-compute/2019.5.0/nv-nsight-cu-cli

## **Collecting Data**

#### By default, results are printed to stdout Use --export/-o to save results to a file, use -f to force overwrite \$ nv-nsight-cu-cli -f -o \$HOME/my report <app>

\$ nv-nsight-cu-cli -f -o \$HOME/my\_report <app
\$ my report.nsight-cuprof-report</pre>

#### Use (env) vars available in your batch script to add report name placeholders

\$ nv-nsight-cu-cli -f -o \$HOME/my\_report\_\${LSB\_JOBID} <app>

\$ my\_report\_951697.nsight-cuprof-report

#### Full parity with nvprof filename placeholders/file macros in next tool version

Disabling PAMI hooks for Spectrum MPI might be required, depending on your application \$ jsrun ... --smpiargs "-disable\_gpu\_hooks" ...

This can be an issue if your application requires \$ jsrun ... --smpiargs "-gpu" ...



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

Data for all processes is stored in one report file

nv-nsight-cu-cli --target-processes all
-o <single-report-name> <app> <args>



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

Ensure that instances don't write to the same report file

nv-nsight-cu-cli -o
report\_\$OMPI\_COMM\_WORLD\_RANK <app>
<args>



Multiple tool instances on the same node are currently not supported

This will be fixed in the next version



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

```
#!/bin/bash
if [[ "$OMPI_COMM_WORLD_RANK" == "3" ]] ; then
    /sw/summit/cuda/10.1.243/ nsight-
compute/nv-nsight-cu-cli -o
report_${OMPI_COMM_WORLD_RANK} --target-
processes all $*
else
    $*
fi
```

## **Retrieving Data**

Use the Nsight Compute CLI (nv-nsight-cu-cli) on any node to import and analyze the report (--import)

More common, transfer the report to your local workstation Reports compress very well, consider tar -czvf before transfer

Reports can be analyzed on any host system (Linux, Windows, Mac) using the local CLI or UI

Analysis in UI is more comprehensive and user-friendly Analysis in CLI is more easily automated (--csv)

### 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

```
cmake/gmxManageNvccConfig.cmake:201
macro(GMX_SET_CUDA_NVCC_FLAGS)
    set(CUDA_NVCC_FLAGS "${GMX_CUDA_NVCC_FLAGS};${CUDA_NVCC_FLAGS};-lineinfo")
endmacro()
```

Source is not embedded in the report, need local access to the source file to resolve in the UI

Comparing different iterations (e.g. optimizations) of the same source file can be difficult Improved in next version to respect file properties

Compiler optimizations can prevent exact source/SASS correlation

#### Source Analysis

#### No -lineinfo

Pag		ourco		ocore:	All			Launch	0000	nlina
ray	e. 🖻			oceas.				Launen.	· pine_s	spinie_e
	Curr	rent 6263	- pme	_spli	Time	: 66,50 L	usecond	Cycles:	83.100	Regs:
	ew:	SASS -								
	28pm	ne_spline_a	nd_sp	read_k	ernellLi4	ELb1ELb	1ELb1EL	b1ELb1E	Lb0EEv2	2PmeG
		ions Everu								
	Juides	Joins Exects								
		Address	Sou	rce						
		0000200		MOV R	1, c[0×	0][0x28				
		0000200		SHFL.	IDX PT,					
		0000200		S2R R	5, SR_0					
		0000200		S2R R	0, SR_0	TAID.X				
		0000200		IMAD						
		0000200		SHF.L	.U32 R6					
		0000200		ISETP	.GE.AND					
	8	0000200								
		0000200		S2R R	4, SR_1	ID.Z				
	10	0000200		S2R R	3, SR_1					
	11	0000200		S2R R	2, SR_1	ID.X				
		0000200		IMAD						
		0000200		IMAD						
	14	0000200		ISETP	.GT.AND					
	15	0000200		IADD3						
	16	0000200		MOV R						
		0000200		IMAD.	WIDE RE					
	18	6666266		LDG.E	SYS R					
	19	6666266		ISETP	.GT.AND					
	20	0000200		SHF.L	.032 RI					
	21	0000200		MOVE						
	22	0000200		IMAD	R8, R5,					
		0000200		IMAD.	WIDE RE		R9, c[0			
	24	0000200		SHF.R	.S32.HI					
	25	6666266		STS [						
	26	0000200		NOP						
	27	0000200		BAR.S	YNC 0x6					
	28	0000200		LUG.E		1 <b>3, (</b> R8)				

#### -lineinfo, unresolved

urrent 6263 - pme_splin Time: 66.24 usecond Cycles: 83 249 Regs: 28 GPU: Tesla V100-5XM21668 SM Frequency: 1.26 cyclensecond CC: 7.0 r: Source and SASS =	Source • Process: All	Launch: - pme_spline_and_spread_kerne	el • Add Baseline • Apply <u>R</u> ules
r: Source and SASS calculate splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_splines.cuh: pmc_calculate_	urrent 6263 - pme_splin Time: 66.24 us	econd Cycles: 83.249 Regs: 28 GPU: Te	sla V100-SXM2-16GB SM Frequency: 1.26 cycle/nsecond CC: 7.0
catcutate splines.cuh       Image: Spline and Spread Remelit4ELDELDELDELDELDELDELDELDELDELDELDELDELDE			
uctions Executed			_Z28pme_spline_and_spread_kernellLi4ELb1ELb1ELb1ELb1ELb1ELb1EL
Source         Live Registers         Sampling Data (All)           i pme_calculate_splines.cuh:         i eleccade.         i eleccoade.         i elecca		E E E E E Resolve	P+ P+
Ipme_spread.cu:         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0 <th0< th="">         0         0</th0<>	# Source pme_calculate_splines.cuh:	Live Registers Sampling Data (All) 6	Instructions Executed 🔹 🔚 📇 🛄 🕅
ism_20_atomic_functions.hpp:         0         7         0002200_0         ISEP.06.4.00 P0, P7, P0, C(b00][0           itexture_indirect_functions.h:         0         0         0002200_0         SER R3, SR, T10.2           0         0000220_0         SER R3, SR, T10.2         0         0         0           0         0000220_0         SER R3, SR, T10.2         0         0         0         0           1         0000220_0         SER R3, SR, T10.2         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0	<pre>pme_spread.cu: cuda_kernel_utils.cuh: cuda_device_runtime_api.h:</pre>		1 6000200. MOV R1, c[0:01](0:02] 2 6000200. eFF SHC1, LKX FT, R2, R2, R2, R2 3 6000200. eFF SHC1, LKX FT, R2, R2, R2, R2 4 6000200. STR R5, SR, CTAID.Y 5 6000200. SFF, LUZ R0, R5, M64, R2
11 0000200. 5/H 60.7 %0 [10.4] FA3 10 000200. 1/M0 F7, R4 ( [0.60] [10.4], FA3 13 0002200. 1/M0 F7, R4 ( [0.60] [10.4], FA3 14 0002200. 1/M0 F7, R47, FA3, FA3, FA3, FA3 15 000200. 1/M1 1/M1 F7, FA3, FA3, FA3, FA3, FA3 15 000200. 1/M1 1/M1 1/M1 1/M1 F7, FA3, FA3, FA3, FA3, FA3, FA3, FA3, FA3	sm_20_atomic_functions.hpp: texture_indirect_functions.h:	0 0 0 0	7 0000200. ISETP.GE.AND P0, PT, R0, c[0x8][0 8 0000200. 0P0 EXIT 9 0000200. SZR R4, SR TID.Z 10 0000200. SZR R3, SR_TID.Y
17 3000200, 0100 1006, 055 812, 160, 000, 07, c15x01[0x23 18 0002200, 0100 105, 655 812, 160, 000, 010 19 0000200, 010 307, 100 307, 100, 002, 102 20 0000200, 010 307, 100, 000, 000, 000, 000, 000, 000 20 0000200, 010 1000, 010, 000, 000, 000			1 11 0000200. SAR R2, SK R10.X [10.X R2] 12 0000200. IMAD R10, R7, R2, (Cdw1[dw1], R3 13 0000200. ISET, CAND, R0, R10, R2 14 0000200. ISET, CAND, R0, R10, R2 15 0000200. g10P M0X R2, R0, R10, R2
22 0000200_010111M00.0126.88, 88, 96, 010, 810 23 0000200_01011M00.0126.88, 88, 96, 016,0161 24 0000200_0157 [613+00.010, 812 25 0000200_010757 [613+00.010, 812 26 0000200_010757 [613+00.01, 812 27 0000000_010757 [613+00.01, 812 27 0000000_010757 [613+00.01, 812 26 000000_010757 [613+00.01, 812 27 0000000_010757 [613+00.01, 812 27 0000000_010757 [613+00.01, 812 27 0000000_010757 [613+00.01, 812 27 000000000000_010757 [613+00.01, 812 27 000000000000000000000000000000000000			17 0000200. @1P0 1MAD.WIDE R6, R6, R7, C[0x0][0x20 18 0000200. @1P0 LDG.E.SYS R12, [R6] 19 0000200. ISET: OF.AND 71, PT, R10, 0x2f, P 20 0000200. @1P1 MV N0, 0x4
			22 0000200. (FF) TMAD R8, R5, 0630, R10 22 0000200. (FF) TMAD R10, R7, R7, 0630, C1AV1[6x1f 24 0000200. (FF) TSG R10, R7, 0617, R12 25 0000200. (FF) STS R15.403/C1, R12 26 0000200. NOP 77 0000000. 010 (SVW 000

#### -lineinfo, resolved

Page: Sourc	te 🔹 Process: All 🔹 Launch: - pme	_spline_and_spread_kerne	al 👻 🗌	Add Baseline 👻 🗸	Apply <u>R</u> ules
Current	: 6263 - pme_splin Time: 66,24 usecond Cycles: 83.2	249 Regs: 28 GPU: Te	sla V10	0-SXM2-16GB SM	Frequency: 1,26 cycle/nsecond CC:
View: Sou	rce and SASS 👻				
pme_spread	1.Cu • E		_228	pme_spline_and_sp	read_kernenci4ELDIELDIELDIELDIELDI
🗏 # Sc	ource	Live Registers 🔺		uctions Executed	
494	float3 atomX;			# Address So	
495	float atomCharge;				MOV R9 Av4
496				2 0000200m @P1	TMAD R8. R5. 0x30. R10
497	const int blockIndex = blockIdx.y * gridD:	10		3 0000200 @IP1	TMAD.WIDE B8. B8. B9. c[0x0][0x
498	const int atomIndexOffset - blockIndex * atomsF	10		4 0000200	SHF.R.532.HI R11, RZ, 0x1f, R10
499				5 0000200 @! P6	STS [R15+0x3c0], R12
500	/* Thread index w.r.t. block */			6 0000200	NOP
501	const int threadLocalid =	15		27 0000200	BAR.SYNC 0x0
502	(threadidx.z * (blockuim.x * blockuim.			8 0000200 @!P]	LDG.E.SYS R13, [R8]
503	/* warp index w.r.t. block - could probably be			9 0000200	LEA.HI R11, R11, R10, RZ, 0x5
> 304	const int warpindex = threadLocatid / warp_size	13		80 0000200	BSSY B0, 0x2008999532c0
505	the datase developed and a second and			81 0000200	LOP3.LUT R6, R4, 0x1, RZ, 0xc0,
500	/* Atom index w.r.t. warp */			2 0000200	SHF.R.S32.HI R7, RZ, 0x5, R11
500	(* Aten index is a black (cheered mereny %)			3 0000200 @!P]	SHF.L.U32 R14, R10, 0x2, RZ
500	<pre>/* Atom index w.f.t. btock/shared memory */ sense int stemEndexLocal - userTedex t stemsDou</pre>			4 0000200	IMAD R10, R3, c[0x0][0x0], R2
519	/* Atom index w s t - slobal memory */			35 0000200	LEA R11, R7, R6, 0x1
511	const int atomIndexGlobal - atomIndexOffset + t			86 0000200	IMAD.WIDE R8, R10, 0x55555556, 1
512	const int atomindexocobat = atomindexorrset + a			37 0000200	ISETP.GT.AND P0, PT, R10, 0x2, I
512	/# Farly roturn for fully omnty blocks at the o			8 0000200	SHF.L.U32 R0, R11, 0x2, RZ
514	* (should only hannen for hillions of input at			9 0000200	LEA.HI R9, R9, R9, RZ, 0x1
515	*/			0 0000200	
516	if (atomIndexOffset >= kernelParams atoms pAtor			1 0000200 @!P]	
517	/			2 0000200	NOP
518				3 0000200	BAR.SYNC 0x0
519				4 0000200 @P0	
528	/t Charges required for both spline and spread			15 0000200	IMAD R8, R9, -0x3, R10
521	if (c useAtomDataProfetch)			6 0000200	IADD3 R9, R10, 0x2, RZ
1				17 0000700	DCCV D1 0v700000057400

# Transitioning from nvprof to Nsight Compute

## nvprof Transition

Check the nvprof (and nvvp) transition guides in the documentation and our blogs <u>https://docs.nvidia.com/nsight-compute/NsightComputeCli/index.html#nvprof-guide</u> <u>https://docs.nvidia.com/nsight-compute/NsightCompute/index.html#nvvp-guide</u> <u>https://devblogs.nvidia.com/migrating-nvidia-nsight-tools-nvvp-nvprof/</u>

Differences	Missing Features (in progress)
New metric names and many more metrics https://docs.nvidia.com/nsight-	Application replay
<u>compute/NsightComputeCli/index.html#nvprof-</u> metric-comparison	No NVLink metrics
	No trace - use <u>Nsight Systems</u>
Cache flush and clock control enabled by default for deterministic data collection	No MPS support
Customizable	

# GROMACS 2020 pme spread/gather Old Version

Memory units more utilized than SM (Compute), but overall utilization is low Nsight Compute hints that this is a latency issue, recommends further sections to check We will still go through other sections for training purposes



Highest utilized pipeline is LSU (Load Store Unit), indicating high load of memory load/store operations



Memory chart shows that stores are much more common in this kernel, transferring ~10x as much data as reads

Since bandwidth is not saturated, it's likely frequent operations



We have many active warps available, but most of them are not eligible (and hence not issued) on average

The next section (Warp State Statistics) can indicate which stall reasons cause this



Most important stall reason (by far) is LG (local/global) Throttle This indicates extremely frequent memory instructions, according to the guided analysis rule



#### The samples locations of those stalls can be looked up on the Source page

Page: Source 👻 Process: All	<ul> <li><u>Launch:</u> pme_spline_ar</li> </ul>	nd_spread_kernel 👻	Add	Baseline 👻 Apply <u>R</u> ules	
Current 6263 - pme_spline_and_spread	Time: 66,24 usecond Cycles	: 83.249 Regs: 28	GPU:	Tesla V100-SXM2-16GB SM F	requency: 1,26 cycle/nsecond CC: 7.0
View: Source and SASS 👻					
pme_spread.cu ×			7	28pme spline and spread ker	helili i4ELb1ELb1ELb1ELb1ELb1ELb1ELb0EEv2
Instructions Executed +					
🗮 # Source	-	Sampling Dat 📤	Ins	structions Executed	- E E E E E E
Select All				# Address Source	
Unselect All				1 0000200 MOV R1,	c[0x0][0x28]
	ckTdx y * aridDim x + b1			2 0000200 @!PT SHFL.ID	X PT, RZ, RZ, RZ, RZ
V #	ckIndex * atomsPerBlock:			3 0000200 S2R R5,	SR_CTAID.Y
-/ Source				4 0000200 S2R R0,	SR_CTAID.X
V Source				5 0000200 IMAD R5	, R5, c[0x0][0xc], R0
Live Registers				6 0000200 SHF.L.U	32 R0, R5, 0x4, RZ
✓ Sampling Data (All)	im.x * blockDim.y)) + (t			7 0000200 ISEIP.0	E.AND PO, PT, RO, CLOXOJLOXITOJ,
Sampling Data (Not Issued)	uld probably be obtained			9 6666266 GFC EXT	SR TTD 7
Instructions Executed	alId / warp_size;			10 0000200 S2R R3.	SR TTD.Y
Predicated-On Thread Instructions Executed				11 0000200 S2R R2.	SR TID.X
Memory Address Space				12 0000200 IMAD R7	, R4, c[0x0][0x4], R3
Memory Access Operation	ad memory #/			13 0000200 IMAD R1	0, R7, c[0x0][0x0], R2
Momony Assocs Size	Index * atomsPerWarp + a			14 0000200 ISETP.G	T.AND PO, PT, R10, 0xf, PT
Memory Access Size	orv */			15 0000200 @!P0 IADD3 F	6, R0, R10, RZ
511 const int atomIndexGlobal = at	omIndexOffset + atomIndex			16 0000200 @!P0 MOV R7,	0x4
512				17 0000200 @!P0 IMAD.WI	DE R6, R6, R7, c[0x0][0x200]
513 /* Early return for fully empt	y blocks at the end			18 0000200 @!P0 LDG.E.S	YS R12, [R6]
514 * (should only happen for bil	lions of input atoms)			19 0000200 ISEIP.G	I.AND P1, PI, R10, 0x2†, PI
515 */				20 0000200 @P0 SHF.L.U	32 RIS, RIO, 0X2, RZ
516 if (atomIndexOffset >= kernelP	arams.atoms.nAtoms)			22 0000200 GIP1 HOV K9,	D5 Av30 D10
517 {				23 0000200 @!P1 IMAD WI	DF R8, R8, R9, c[0x0][0x1f8]
518 return;				24 0000200 SHF.R.S	32.HI R11. RZ. 0x1f. R10
519 }				25 0000200 @!P0 STS [R]	5+0x3c0], R12
520 /* Charges, required for both	spline and spread */			26 0000200 NOP	
521 IT (C_USEALONDATAPTETETCh)				27 0000200 BAR SVN	C AVA

ome_cal	culate_splines.cuh 👻 🖻	P+ P+ stall_lg		- 12 2 4
	Source		Sampling Data (All)	stall_lg
197 198	<pre>sm_fractCoords[sharedMemon tableIndex += tInt;</pre>	ryIndex] = t - tInt;	18]	
200	<pre>assert(tint &gt;= 0); assert(tint &lt; c_pmeNeighbourge)</pre>	orUnitcellCount * n);	U	
202 203	<pre>// TODO have shared table // TODO compare texture/LD</pre>	for both parameters to OG performance		
204 205	sm_fractCoords[sharedMemor fetchFromParamLook	ryIndex] += <pre>kupTable(kernelParams.c</pre>		
206 207	<pre>sm_gridlineIndices[shared</pre>	kernelParams.f MemoryIndex] =		
208 209	fetchFromParamLook	<pre>kernelParams.c kernelParams.c</pre>		
210 211	if (writeGlobal) {		0	
212	gm_gridlineIndices[atc	omIndexOffset * DIM + s	282	224
213 214	sm_gridlineInd }	lices[sharedMemoryInde>		otal Sample Count: 282 spatch Stall: 1 ( 0.4%)
215 216			Lg M	Throttle: 224 (79.4%) ath Pipe Throttle: 1 ( 0.4%)
217 218	<pre>/* B-spline calculation */</pre>		No	ot Selected: 12 ( 4.3%) elected: 1 ( 0.4%)
219 220	<pre>const int chargeCheck = pme_g if (chargeCheck)</pre>	pu_check_atom_charge(a	0	

Disabling global memory writes to store temporary data (for the gather kernel) could reduce this latency issue

This implies that the gather kernel has to re-compute this data

pn	ne_cal	culate_spli	nes.cuh 🔻 🖯		stall_lg			-	æ 4	
	#	Source				Sampling Data (All)	stall lg			
	197		<pre>sm_fractCoords[share</pre>	dMemoryIndex] = t ·	tInt;	18	0			
	198		<pre>tableIndex += tInt;</pre>			4	Θ			
	199		<pre>assert(tInt &gt;= 0);</pre>		80.	0				
	200		<pre>assert(tInt &lt; c_pmeN</pre>	leighborUnitcellCour	it * n);	0				
	201					0				
	202		// TODO have shared t	table for both para	meters to	0				
	203		<pre>// TODO compare text</pre>	ure/LDG performance	S.	0				
	204		<pre>sm_fractCoords[share</pre>	dMemoryIndex] +=	0.0	0				
	205		fetchFromPara	amLookupTable(kerne	lParams.	0				
	206			kerne	lParams.f	0				
	207		sm_gridlineIndices[s	haredMemoryIndex] =		0				
	208		fetchFromPara	amLookupTable(kerne	lParams.g	0				
	209			Kerne	Carams . g	2				
	210		if (writeGlobal)			0				
	211		1			0				
	212		gm_gridlineIndice	es[atomIndexOffset	* DIM + s	282	224			
	213		sm_gridu	ineindices[sharedMe	moryinde>	T	otal Samp	le Count:	282	
	214		1			D	ispatch St	all: 1 ( 0.4	1%)	
	215	4					a in Pige T	224 (79.4	4%)	%)
	210		* D coline colculation :	*1		M	io Throttle	: 43 (15.	2%)	
	21/		B-sprine carculation			N	ot Selecte	d: 12 ( 4.	3%)	
	210		onst int charge(back -	nme anu check atom	charge (a	5	elected: 1	(0.4%)		
	229	-	f (chargeCheck)	hue_dha_cueck_arom	citar ge (a	0				
	220	1	f (chargeCheck)			0				

### gather: Old Version (overview)

## gather: Old Version

#### More balances compute/memory utilization, but also likely latency bound



### gather: Old Version

Reads temporary spline\_and\_spread kernel data from global memory Therefore, much more load operations and data transfered in that direction



### gather: Old Version

#### Long Scoreboard stalls cause most wasted cycles These indicate waiting on local or global memory



# GROMACS 2020 pme spread/gather New Version

#### **Code Changes**

https://redmine.gromacs.org/projects/gromacs/repository/revisions/22118220401cee6f51d49c0a034e9fe5b4ba4260/diff?utf8=%E2 %9C%93&type=sbs

Two new template arguments added to spread/gather kernels Optimal kernel selected based on input data size Disabled temp data storage in global memory for this analysis

pme_spline_and_spread_kernel	pme_gather_kernel
<b>writeSplinesToGlobal</b>	<b>readGlobal</b>
control if we should write spline data to	control if we should read spline values
global memory	from global memory
useOrderThreadsPerAtom*	<i>useOrderThreadsPerAtom</i> *
control if we should use order or	control if we should use order threads per
order*order threads per atom	atom (order*order used if false)

\* not activated

#### Overall performance improvement is ~15% (fewer cycles) Highest contributor appears to be the 54% reduced GPU DRAM throughput (SOL FB)



Compute Workload Analysis shows slightly reduced usage of the load-store units pipeline in exchange for increased utilization of arithmetic pipelines (ALU, FMA)



#### Reduced global store requests and data transfers to device memory



#### The eligible and issued warps/scheduler improved slightly (but are still quite low)



To increase the number of eligible warps either increase the number of active warps or reduce the time the active warps are stalled.

The improvement is due to reduced LG (local/global) Throttle stalls (since we have fewer writes to memory) Could be further reduced in a follow-up optimization



#### Performance decreased slightly compared with "unoptimized" version The other individual sections allow us to identify what has changed in detail



Recomputing instead of reading from global memory shows reduced cycles/inst for Long Scoreboard stalls...

...which translates to improved eligible and issued warps per scheduler



#### While the kernel executes instructions more efficiently now (higher IPC)...

Compute Workload Analysis					Ω
Detailed analysis of the compute resources of the streaming multiprocessors very high utilization might limit the overall performance.	(SM), incl	luding the a	chieved instructions per clock (IPC) and the utilization of each available pip	eline. Pipe	lines with
Executed Ipc Elapsed [inst/cycle]	2,69	(+33,26%)	SM Busy [%]	75,08	(+28,69%)
Executed Ipc Active [inst/cycle]	2,98	(+28,55%)	Issue Slots Busy [%]	75,08	(+28,69%)
Issued Ipc Active [inst/cycle]	3,00	(+28,69%)			

#### ... it also executes a lot more instructions in total (to re-compute values instead of loading them)

Instruction Statistics					Q
Statistics of the executed low-level assembly instructions (SASS). The in implies a dependency on few instruction pipelines, while others remain 'Executed Instructions' are measured differently and can diverge if cycle	struction mix unused. Using s are spent ir	provides ins multiple pip system call	ight into the types and frequency of the executed instructions. A narro elines allows hiding latencies and enables parallel execution. Note tha s.	w mix of instruc t 'Instructions/C	tion types pcode' and
Executed Instructions [inst] Issued Instructions [inst]	7.104.000 7.160.121	(+48,93%) (+49,09%)	Avg. Executed Instructions Per Scheduler [inst] Avg. Issued Instructions Per Scheduler [inst]	22.200 22.375,38	(+48,93%) (+49,09%)

## gather: Source Analysis

On the collapsed Source page, we can quickly identify where the new instructions originate

#### Old

pme_calculate_splines.cuh 🝷 🕀	P+ P+	Instructions Executed	- <u>(</u> =	E E E E
# Source	Sampling Data (All)	Sampling Data (Not Issued)	Instructions Executed	Predicated-On Thread Instructions Executed
1 pme_calculate_splines.cuh	0	Ð		
2 pme_gather.cu	974	363	4.434.000	115.680.000
! cuda_kernel_utils.cuh	0	<u>e</u>		
<pre>! cuda_device_runtime_api.h</pre>	0	0		
sm_30_intrinsics.hpp	0	0		
texture_indirect_functions.	Θ	0		

#### New

pme_calculate_splines.cuh 👻 臣	P+ P+ I	nstructions Executed	- î= î=	2222
# Source	Sampling Data (All)	Sampling Data (Not Issuec)	Instructions Executed	Predicated-On Thread Instructions Executed
1 pme_calculate_splines.cuh	320	57	2.088.000	14.784.000
2 pme_gather.cu	690	146	4.488.000	113.040.000
! cuda_kernel_utils.cuh	0	<del>0</del>		
<pre> cuda_device_runtime_api.h</pre>	0	0		
sm_30_intrinsics.hpp	Θ	0		
texture_indirect_functions.	Θ	0		

### **New Version Summary**

#### Overall, combined performance improved by ~10% Use CSV export from CLI or UI to further analyze data in e.g. Excel

Function Name	Dema Proce: Device I Cycles	[cycle] 9	OL Memory [%]	SOL SM [%]	Function Name	Demanc Proce: Device Cycle	[cycle]	SOL Memory [%]	SOL SM [%]
pme spline and spread kernel	voi… [17… Tesla…	83.249	49,04	20,33	pme spline and spread kernel	void … [13… Tesl…	69.516	44,82	21,65
pme gather kernel	voi… [17… Tesla…	29.646	53,38	50,84	pme gather kernel	void … [13… Tesl…	33.206	60,90	67,82
pme spline and spread kernel	voi… [17… Tesla…	83.236	49,06	20,32	pme spline and spread kernel	void … [13… Tesl…	68.969	45,18	21,84
pme gather kernel	voi… [17… Tesla…	29.890	53,01	50,37	pme gather kernel	void … [13… Tesl…	33.497	60,29	67,15
pme spline and spread kernel	voi… [17… Tesla…	83.624	48,80	20,23	pme spline and spread kernel	void … [13… Tesl…	68.516	45,44	21,97
pme gather kernel	voi… [17… Tesla…	29.051	54,58	51,86	pme gather kernel	void … [13… Tesl…	33.459	60,44	67,31
pme spline and spread kernel	voi… [17… Tesla…	83.617	48,80	20,23	pme spline and spread kernel	void … [13… Tesl…	69.197	45,00	21,75
pme gather kernel	voi… [17… Tesla…	29.636	53,52	50,81	pme gather kernel	void … [13… Tesl…	34.020	59,38	66,15
pme spline and spread kernel	voi… [17… Tesla…	83.070	49,12	20,37	pme spline and spread kernel	void … [13… Tesl…	70.337	44,29	21,39
pme gather kernel	voi… [17… Tesla…	29.591	53,68	50,87	pme gather kernel	void … [13… Tesl…	33.389	60,50	67,42
pme spline and spread kernel	voi… [17… Tesla…	83.235	49,05	20,32	pme spline and spread kernel	void … [13… Tesl…	69.615	44,73	21,62
pme gather kernel	voi… [17… Tesla…	29.705	53,20	50,67	pme gather kernel	void … [13… Tesl…	33.558	60,23	67,06
pme spline and spread kernel	voi… [17… Tesla…	82.910	49,19	20,40	pme spline and spread kernel	void … [13… Tesl…	69.261	44,96	21,73
pme gather kernel	voi… [17… Tesla…	29.233	54,32	51,50	pme gather kernel	void … [13… Tesl…	33.457	60,46	67,32
pme spline and spread kernel	voi… [17… Tesla…	83.109	49,09	20,35	pme spline and spread kernel	void … [13… Tesl…	68.927	45,16	21,84
pme gather kernel	voi… [17… Tesla…	29.634	53,61	50,79	pme gather kernel	void … [13… Tesl…	33.874	59,62	66,40
pme spline and spread kernel	voi… [17… Tesla…	83.299	48,98	20,31	pme spline and spread kernel	void … [13… Tesl…	69.266	44,95	21,73
pme gather kernel	voi… [17… Tesla…	30.130	52,62	49,97	pme gather kernel	void … [13… Tesl…	33.221	60,91	67,82
pme spline and spread kernel	voi… [17… Tesla…	83.595	48,84	20,24	pme spline and spread kernel	void … [13… Tesl…	68.452	45,50	22,00
pme gather kernel	voi… [17… Tesla…	29.554	53,74	50,99	pme gather kernel	void … [13… Tesl…	33.432	60,54	67,32

## Customize Data Collection and Analysis

#### **Customize Sections**

```
Identifier: "SpeedOfLight"
DisplayName: "GPU Speed Of Light"
Description: "High-level overview of ..."
Order: 10
Sets {
  Identifier: "default"
Sets {
  Identifier: "full"
Header {
  Metrics 4
    Label: "SOL SM"
    Name: "sm throughput.avg.
pct of peak sustained elapsed"
  Metrics {
    Label: "Duration"
    Name: "qpu time duration.sum"
  Metrics {
    Label: "SOL Memory"
    Name: "gpu compute memory throughput.avg.
pct of peak sustained elapsed"
  Metrics {
    Label: "Elapsed Cycles"
    Name: "gpc cycles elapsed.max"
```

. . .



Metrics collection Metric presentation Tables Charts Source page correlation Details page ordering Section set association

#### **Customize Rules**

```
import NvRules
import math
                                                                                                              Recommendations
def get identifier():
                                                                          [Warning] Compute is more heavily utilized than Memory: Look at `Compute Workload Analysis` report section to see what the compute
    return "SOLBottleneck"
                                                                Bottleneck
                                                                         doing. Also, consider whether any computation is redundant and could be reduced or moved to look-up tables.
def get section identifier():
    return "SpeedOfLight"
                                                                           Python rules accessing collected data
def apply(handle):
    ctx = NvRules.get context(handle)
                                                                           Available in UI and CLI
    action = ctx.range by idx(0).action by idx(0)
                                                                           Performance analysis guidance
    fe = ctx.frontend()
                                                                                           By NVIDIA
    num waves =
action.metric by name("launch waves per multiprocessor")
                                                                                           By your own experts
.as double()
    smSolPct = action.metric by name("sm throughput.avg
                                                                           Allow users better understanding of recommendations
.pct of peak sustained elapsed").as double()
    memSolPct =
action.metric by name("gpu compute memory throughput.avg
                                                                           In next version: dynamic report navigation
.pct of peak sustained elapsed").as double()
    balanced threshold = 10
    latency bound threshold = 60
    no bound threshold = 80
    waves threshold = 1
```

•••

# Conclusion

#### **Known Issues/Outlook**

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

Outlook for next version

Improved multi-process/MPI support

Parity with nvprof report name placeholders (process ID, env var, running number)

Better kernel name demangler

Improved memory workload analysis tables

Dynamic report navigation

Uncoalesced memory rules

#### Conclusion

Nsight Compute enables detailed kernel analysis

Rules give guidance on optimization opportunities and help metric understanding

Limit sections/metrics to what is required when overhead is a concern

Still requires level of hardware understanding to fully utilize the tool - pay attention to rule results



# THANK YOU!

Download	https://developer.nvidia.com/nsight-compute (can we newer than toolkit version)
Documentation	https://docs.nvidia.com/nsight-compute (and local with the tool)
Forums	https://devtalk.nvidia.com
Further Training	Blue Waters Seminar <u>https://bluewaters.ncsa.illinois.edu/webinars/petascale-</u> <u>computing/nsight-compute</u> GTC 2019
	https://developer.nvidia.com/gtc/2019/video/S9345
	Blog posts
	https://devblogs.nvidia.com/using-nsight-compute-to-inspect-your-kernels/
	https://devblogs.nvidia.com/migrating-nvidia-nsight-tools-nvvp-nvprof/

