

GPU COMPUTING 3 - TOOLS

Mathias Wagner, Lattice Practices 2024

OVERVIEW CUDA Tools

CUDA-GDB

Extension of GDB, allows debugging of CUDA application

COMPUTE-SANITIZER

Valgrind like tool to check functional correctness

NSIGHT PROFILERS

Timing of Kernel and API calls (Timeline in GUI)

Detailed kernel metrics and events Guided analysis for performance optimization

DEBUGGING

Debugging Correctness: Best Practices

Before you start

- Crashes are "nice" the stacktrace often points to the bug
- Prerequisite: Compile flags
 - While developing, always use -g -lineinfo
 - Use g G for manual debugging
 - Specific flags for compilers/languages (e.g. gfortran): -fcheck=bounds
- Memory corruption: Out-of-bounds accesses may or may not crash
 - compute-sanitizer: Automate finding these errrors
- Other issues: Manual debugging
 - *cuda-gdb*: Command-line debugger, GPU extensions
 - CUDA_LAUNCH_BLOCKING=1 forces synchronous kernel launches

compute-sanitizer

Functional correctness checking suite for GPU <u>https://docs.nvidia.com/compute-sanitizer/ComputeSanitizer/</u>

- compute-sanitizer is a collection of tools
- memcheck (default) tool comparable to Valgrind's memcheck.
- Other tools include
 - racecheck: shared memory data access hazard detector
 - initcheck: uninitialized device global memory access detector
 - synccheck: identify whether a CUDA application is correctly using synchronization primitives
- Main usage: Auto-detect invalid GPU code and shortcut debugging effort
 - Directly pinpoint source code line/addresses, access size
- Leak-checking for device allocations forgot to call cudaFree()?
 - --leak-check full
- Filtering and other capabilities. Two commonly useful switches:
 - --log-file output.log
 - Separates (potentially verbose) output into separate file
 - --kernel-regex kns=some_substring
 - Only checks kernels containing "some_substring"

cuda-gdb

Extends GDB for CUDA applications https://docs.nvidia.com/cuda/cuda-gdb/index.html

- "Symbolic Debugger" leaverage debug symbols to correlate execution issues with original source code
- Interactive/manual tool, with useful shortcuts
 - https://docs.nvidia.com/cuda/cuda-gdb/index.html#automatic-error-checking
- Textual, like a shell for debugging Not the easiest to master, but very powerful, and works everywhere
- Basic workflow for segfaults
 - Crashing app invoked via
 - ./my_app_name my_app_arg another_arg
 - becomes
 - cuda-gdb --args ./my_app_name my_app_arg another_arg
 - Shows you the debugger shell prompt: (cuda-gdb)
 - Launch program with "run"
 - Identify the segfault Done ^(C)
- Advanced workflow to step through execution, understand program flow, inspect and modify variables,...

cuda-gdb Cheat Sheet

(doubles as a GDB cheat sheet)

- Most commands have abbreviations
 - continue \rightarrow cont, break \rightarrow b, info \rightarrow i, backtrace \rightarrow bt, ...
 - cuda thread 4 \rightarrow cu th 4
- Use TAB completion to help you remember command names
- Use help and apropos to avoid a round-trip to the browser (try: apropos cuda.*api)

run	Begin progam execution under debugger					
backtrace Print call stack (e.g. after an exception)						
list	List source code around current location					
print <mark><var></var></mark>	Print contents of <mark><var></var></mark> , e.g. "print i" to print the loop counter i					
set var <mark><var></var></mark> = <mark><value></value></mark>	Set value of < <u>var></u> to < <u>value></u> , e.g. "set var i=42"					
break 10 break foo.cpp:10 break my_func	Set breakpoint (suspend execution) on: line 10 in current file line 10 in file foo.cpp function my_func in any file					
set cuda api_failures <mark>stop</mark>	Break on any CUDA API failures (e.g. launch errors)					
continue / next / step	Resume execution (after hitting breakpoint) until next: break / line / instruction					
info locals	Print all local variables in current scope					
info cuda threads	Print current thread configuration					
cuda thread <mark>15</mark>	Switch focus to thread (here: 15)					

The Most Essential Command

In case of segfault, remember the backtrace

• If your app crashes or terminates unexpectedly, the debugger can very often tell you the exact location of the issue

```
• Both in CPU and GPU code
```

```
$ cuda-gdb --args ./gpu-print
(cuda-gdb) run
[...]
CUDA Exception: Warp Illegal Address
The exception was triggered at PC 0xacbc90 (gpu_print.cu:19)
Thread 1 "gpu_print" received signal CUDA_EXCEPTION_14, Warp Illegal Address.
[Switching focus to CUDA kernel 0, grid 1, block (0,0,0), thread (0,0,0), device 0,sm 0,warp
0,lane 0]
0x000000000acbca0 in print_test<<<(2,1,1),(32,1,1)>>> () at gpu_print.cu:19
19 double x = *(double*)nullptr;
(cuda-gdb) bt # "backtrace"
#0 0x00000000acbca0 in print_test<<<(2,1,1),(32,1,1)>>> () at gpu_print.cu:19
```

- Backtrace tries to print all stack frames (i.e. function calls) with line information up to the current location
 - Equally useful when manually debugging or using breakpoints
 - Some errors can corrupt the stack, making the backtrace less useful

GPU-Specifics

New commands in cuda-gdb

• GPU-specifics: Setting the *focus*

```
(cuda-qdb) i cuda threads
 BlockIdx ThreadIdx To BlockIdx ThreadIdx Count
                                                        Virtual PC
                                                                       Filename Line
Kernel 0
*
  (0, 0, 0)
                                            32 0x00000000000acbf90 gpu_print.cu
          (0,0,0)
                       (0,0,0) (31,0,0)
                                                                                   19
                       (1,0,0) (31,0,0)
          (0, 0, 0)
   (1,0,0)
                                             32 0x00000000000acbf60 gpu_print.cu
                                                                                   18
(cuda-qdb) cuda thread
thread (0,0,0)
(cuda-gdb) cuda thread 10
[Switching focus to CUDA kernel 0, grid 1, block (0,0,0), thread (10,0,0), device 0,sm 0,warp 0,lane
10]
19
               printf("blockIdx.x = %d, threadIdx.x = %d, i = %d\n", blockIdx.x, threadIdx.x, i);
```

- Focus can be set to specific blocks, SMs, devices, ... help cuda
 - Hardware and software abstractions (e.g. blocks vs. SMs)
- Options: Try (cuda-gdb) set cuda<ENTER> for a list
 - Two commonly-used options: api_failures and launch_blocking

NVIDIA NSIGHT SUITE

PERFORMANCE OPTIMIZATION

What exactly is the performance bottleneck?

You might have a feeling where your application spends most of it's time ...

... but a more analytic approach might be better

... but keep in mind that you might kill some cats in the process

(Profiling creates overhead)

WHAT DOES A PROFILER DO?

Sampling vs. Instrumentation (very simplified)



Every ms, take a sample of callstack

while () {	Samples
do_nothing()	0
intense_calculation()	23
sleep() }	12

(+) Hot spots show up, low overhead

(-) May miss some calls

ク



while (...) {

trace_do_nothing() -> do_nothing()

trace_intense_calculation() -> intense_calculation()

```
trace_sleep() -> sleep()
```

(+) Captures whole program, full call chains

(-) Potentially higher overhead, skew

THE NSIGHT SUITE COMPONENTS

How the pieces fit together





WHETTING YOUR APPETITE

Timeline overview in Nsight Systems GUI



A FIRST (I)NSIGHT Maximum achievable speedup: Amdahl's law

Amdahl's law states overall speedup s given the parallel fraction p of code and number of processes N

$$s = \frac{1}{1 - p + \frac{p}{N}} < \frac{1}{1 - p}$$

Limited by serial fraction, even for $N \rightarrow \infty$

Example for p = 30%

Also valid for per-method speedups





A FIRST (I)NSIGHT

Recording with the GUI

🕐 mhrywniak@localhost:8200 🔹 🎸 🍋 🕿 🛩 Target is ready More info		Connect directly
▼ ✓ Sample target process	Start	, ,
Sampling Period: 1,000,000 events The sampling period is the number of 'CPU Instructions Retired' events counted before a CPU instruction pointer (IP) sample is collected. If configured, call stacks may also be collected. The smaller the sample period, the higher the sampling rate. Lower sampling periods will increase overhead and significantly increase the size of result file(s). V Collect call stacks of executing threads Target application	Start profiling manually Start profiling after 10,0 ♀ Limit profiling to 10,0 ♀ Hotkey {F12} Start/Stop (not available in console apps)	Or use an SSH Tunnel: ssh -L 8200:compute- node:22 login-node
Mode: Specify process launch options below Command line with arguments: Edit arguments /gpfs/fs1/mhrywniak/code/lbmd2q37/d2q37-v1-acc-mpi Working directory: /gpfs/fs1/mhrywniak/code/lbmd2q37 Environment variables Trace fork before exec 	Manage targets × Recent connections Make sure your target is on the same subnetwork as this computer. Target Username Membrywniak@localhost:8200 mhrywniak	Select traces to collect
 ✓ Collect CPU context switch trace ✓ Collect OS runtime libraries trace ✓ Collect CUDA trace Collect OpenMP trace Collect GPU context switch trace ✓ Collect MPI trace ✓ Collect MPI trace ✓ Collect MPI trace ✓ Collect NVTX trace 	Le Create a new connection X Delete Z Edit Connect Close	

A FIRST (I)NSIGHT

Recording an application timeline

1) We'll use the command line

```
mpirun -np $NP \
nsys profile --trace=cuda,nvtx,mpi \
--output=my_report.%q{OMPI_COMM_WORLD_RANK}.qdrep ./myApp
```

Note: Slurm users, try srun ... %q{SLURM_PROCID}

2) Inspect results: Open the report file in the GUI

Also possible to get details on command line (documentation), nsys stats --help

See also https://docs.nvidia.com/nsight-systems/, "Profiling from the CLI on Linux Devices"

USING NSIGHT SYSTEMS

Recording with the CLI

Use the command line

srun nsys profile --trace=cuda,nvtx,mpi --output=my_report.%q{SLURM_PROCID} ./jacobi -niter 10

- Inspect results: Open the report file in the GUI
 - Also possible to get details on command line
 - Either add --stats to profile command line, or: nsys stats --help
- Runs set of reports on command line, customizable (sqlite + Python):
 - Useful to check validity of profile, identify important kernels

Ru	ning	[/reports/gp	oukernsum.p	y jacobi_n	metrics_mor	re-nvtx.0.	<pre>sqlite]</pre>	•	
Ti	me(%)	Total Time (ns)	Instances	Avg (ns)	Med (ns)	Min (ns)	Max (ns)	StdDev (nsi	Name
	99.9	36750359 22816	20 2	1837518.0 11408.0	1838466.5 11408.0	622945 7520	3055044 15296	1245121 7 5498.	void jacobi_kernel initialize_boundaries

LOOKING AT A SIMPLE EXAMPLE

🚯 NVIDIA Nsight Systems 2020.5.1 📃 🗮	: 🖂 🖸 🔌		_ ¤ ×
<u>F</u> ile <u>V</u> iew <u>T</u> ools <u>H</u> elp			
Project Explorer ×	Project 1 × scale_report.qdrep ×		
Scale report.gdrep	■ Timeline View -		▲ <u>4 warnings, 15 messages</u>
	0s • m	s +50ms +100ms +150ms +200ms +250ms +300ms +350ms	+400ms +450ms +500ms
	▶ CPU (96)		
	CUDA HW (A100-SXM4-40GB)		
	 Threads (7) 		
	▼ 🗹 [30604] scale_vector_u >		
	OS runtime libraries	and the local sectors	se
	CUDA API	cudaMallocManaged	
	Profiler overhead	OS ru OpenGL	C
	o threads hidden – +	· · · · · · · · · · · · · · · · · · ·	
			-
	4		•
	Events View 👻		
		Na	ame 👻 🔍
			Description:
		Right-click a timeline row and select "Show in Events View" to see events here	

USING CALLSTACK SAMPLES

v [50004] Scale	<u>R</u> emove Filter	
OS runtime libr	Undo Zoom (0)	Backspace
CUDA API	Reset Zoom	
Profiler overhea	Pin row	Ctrl+P
6 threads hidder	Show in Events View	

Events View makes information searchable

"Highlight All" shows all matches

Can search in description, includes callstack

h	View 🔻		A warnings, 15 messages
	0s •	ns +100ms +200ms +300ms	+/00m +500ms
 CPU (96) 			
CUDA HW	(A100-SXM4-40GB)		
 Threads (7 	7)		
▼ 	4] scale_vector_i +		
OS run	time libraries		
CUDA /	API	cudaMallocManaged	
Profiler	overhead	O Ope CUDA	
6 thread	shidden -+		
		4	
Events View	•	1 of 14 matches Description	
Events View	•	1 of 14 matches Description std::abs	Description:
Events View	• Name scale vector um	1 of 14 matches Description std::abs	Description: Sampling point
Events View •	Name scale_vector_um scale vector um	1 of 14 matches Description std::abs	
Events View .02 .03 .04	Name scale_vector_um scale_vector_um scale_vector_um	1 of 14 matches Description std::abs	Call stack at 0.40045s
Events View .02 .03 .04 .05	Name scale_vector_um scale_vector_um scale_vector_um scale_vector_um scale_vector_um	1 of 14 matches Description std::abs n!std::abs() n!std::abs() n!std::abs()	Call stack at 0.40045s Std::abs()
Events View .02 .03 .04 .05 .06	Name scale_vector_um scale_vector_um scale_vector_um scale_vector_um scale_vector_um scale_vector_um	1 of 14 matches Description std::abs n!std::abs() n!std::abs() n!main n!std::abs()	Call stack at 0.40045s Sampling point Call stack at 0.40045s State votor um! Std::abs() Std::abs()

ADDING SOME COLOR

Code annotation with NVTX

Like manual timing, only less work

Nesting, timing

Correlation, filtering

Timeline View	•					🖾 Q 1x	I I I I	<u>∧</u> <u>3 warnings,</u> 1	.7 messag
	0s -	+100m	ns +1	.50ms	+200r 219,3ms	+250ms	+300ms	+350ms +4	00ms
CPU (96)									
CUDA HW (A100-S	XM4-40GB)								
Threads (7)									
▼ 🗹 [31520] scale_	vector_um +								
OS runtime libr	aries				ioctl		J. Jakilalalalalalalalala	sem	s
			vm 11			main [319,944 ms]		
NVTX	×*				ini	t [303,215 ms]	L [202.407]		v
						mai	loc [202,197 ms]		
CUDA API		0.01	CUDA CI:		-	CUO	amallocmanaged		
Profiler overnea	ad	OpenGL	CUDA profiling	g initi					
6 threads hidden	– +	•							
									•
		•							۱.
Events View 💌									
	J						Name 💌		
£ 🔺	Name						Start	Description:	
1	🔻 🛛 main						0,0941701s	kernel	
2	▶ ir	nit					0,0941715s	Begins: 0,397388s Ends: 0,401767s (+	4,379 ms)
4	k	ernel					0,397388s	Thread: 31520	
-		-1:							

ADDING NVTX

Simple range-based API

#include <nvToolsExt3.h>

Copy&paste PUSH/POP macros (or module)

PUSH(name, color)

Sprinkle them strategically through code

NVTX v3 is header-only

Not shown: Advanced usage (domains, ...)

https://github.com/NVIDIA/NVTX

```
int main(int argc, char** argv){
    PUSH("main", 0)
    PUSH("init", 1)
```

POP PUSH(<mark>"kernel",</mark> 2)

```
scale<<<gridDim, blockDim>>>(alpha, a, c, m);
```

```
cudaDeviceSynchronize();
POP
```

```
PUSH("validate", 3)
```

https://developer.nvidia.com/blog/cuda-pro-tip-generate-custom-application-profile-timelines-nvtx/

https://developer.nvidia.com/blog/customize-cuda-fortran-profiling-nvtx/

ZOOMING IN

Regions of interest

activity CUDA HW (A100-SXM4-40GB) TTUTTI N 1000 THE P 64.1% Context 1 **Kernel** 100.0% Kernels scale main [4,301 ms] NVTX kernel [4,301 ms] 35.9% Unified memory Remark of same bound by same the 100.0% Memory 62.9% HtoD transfer anto la Circle II l. 37.1% DtoH transfer Threads (7) ▼ 🗹 [31520] scale_vector_um -111 OS runtime libraries sem timedwait main [319,944 ms] init [303,215 ms] kernel [4,379 ms] validate [10,543 ms] NVTX malloc [202,197 ms] cudaMallocManaged cudaDeviceSynchronize cud...] c... CUDA API

GPU

Kernel launch

UM migrations and page faults

Use Amdahl's law as heuristic

MINIMIZING PROFILE SIZE

Shorter time, smaller files = quicker progress

Only profile what you need - all profilers have some overhead

Bonus: lower number of events => smaller file size

Add to nsys command line:

--capture-range=nvtx --nvtx-capture=any_nvtx_marker_name \
--env-var=NSYS_NVTX_PROFILER_REGISTER_ONLY=0 --kill none

Alternatively: cudaProfilerStart() and -Stop()

--capture-range=cudaProfilerApi

THEFT		• T 🔤 T	1 11 1
			100
	1110 00 10	494 184	8-011-04 (44
	kernel_	and_validate [39.7	/03 ms]
kernel [7.01	vali	idate [23.668 ms]	
sc cudaDe			
JICI			

OTHER FEATURES

We only covered a small subset

"Traditional" top-down or bottom-up stack views

Lots of different traces (MPI, OpenACC, OpenMP, ...)

Data export (csv, sqlite, ...)

Customizable reports via Python scripts

Full guide:

https://docs.nvidia.com/nsight-systems/UserGuide

Top-Down View Process [31520] scale_vector_um (7 of 7 threads)							
Tilter 901 samples are used.							
Symbol Name Self, % otal, 9 🕶 N							
▼_start		5 <mark>4,61</mark> /					
▼libc_start_main		5 <mark>4,61</mark> /					
✓ main	7,66	54,38 /					
cudaError cudaMallocManaged <float>(flo</float>		22,4 <mark>2</mark> /					
cudaSetDevice		19,3 <mark>1</mark> /					

[hrywnia	hrywniak1@jwlogin24 task3]\$ nsys stats scale_report.qdrep									
Using sc Running	#Sing scate_report.squite for sut queries. Running [//software/iuwelsbooster/stages/2020/software/Nsight-Svstems/2020.5.1-GCCcore-9.3.0/target-linux-x64/reports/cudaap									
isum.py	isum.py scale_report.sqlite]									
Time(%)	Total Time (ns)	Num Calls	Average	Minimum	Maximum	Name				
	470.455.6		4704556 0	4704550	4704550					
67.5	4/04556		4704556.0	4704556	4704556	cudaDeviceSynchronize				
32.5	2265468		2265468.0	2265468	2265468	cudaLaunchKernel				
Punning	[/n/coftware/juwo]	shoostor/sta	agos /2020 /	oftware/	leight-Suc	$t_{oms}/2020 = 5 + 1 - 6 + C + c_{oms} - 0 = 3 + 0 + t_{argot} - 1 + t_{argot} - 2 + 0 + t_{argot} + 1 + t_{argot} + 0 + t_{$				
	scale report solit		ages/2020/s	sortware/r	vsight-sys					
nsumpy	scare_report.sqtrt									
Time(%)	Total Time (ns)	Instances	Average	Minimum	Maximum	Name				
100.0	4709010	1	4709010.0	4709010	4709010	scale(float, float*, float*, int)				
Running timesum.	kunning [/p/software/juwelsbooster/stages/2020/software/Nsight-Systems/2020.5.1-GCCcore-9.3.0/target-linux-x64/reports/gpumem timesum.py scale_report.sqlite]									
Time(%)	Total Time (ns)	Operations	Average	Minimum	Maximum	Operation				
65 7	1706510	164	2050 2	2020	22000	[CUDA Unified Memory memory Htep]				
24.2	1/80518	464	0720 1	2039	52800	[CUDA Unified Memory memory http://				
34.3	934091	96	9730.1	2111	55119	[CODA ONITIEd Memory memopy DCOH]				

WHEN TO MOVE ON

Proper tool for the job

Specialized MPI profiling/bottlenecks, load imbalance

Kernel-level profiling -> Nsight Compute

Used later on (get the low-hanging fruit first!)

Use it when you find a hotspot kernel

SUMMARY

How to approach porting your own code

Start with Nsight Systems and record a first profile

Identify roughly some features (use call stacks, code knowledge), add NVTX

Add and customize traces as needed

Use capture ranges

Iteratively eliminate "blank" spots - is the GPU active?

Switch to more specialized profilers as needed



DRILLING DOWN ON A KERNEL

Analysis with Nsight Compute

		C	Analyze the selected kernel with NVIDIA Nsight Compute
Right-click menu in Nsight Systems	100.0% collide(double* double const* part const* int)		Copy Current Time
set command line			<u>R</u> emove Filter
			Undo Zoom (11)

Run command line

ncu --page details --import-source true --set full \
 -k collide -s 3 -c 1 -f -o my report ./lbmD2Q37

Important switches for metrics collection, pre-selected sets

```
Fully customizable, ncu --help. Check --list-metrics and --query-metrics
```

Here: profile with CLI, use GUI for analysis and load report file

Alternatively, interactive analysis of application through GUI. API Stream can be very useful.

Nsight Compute GUI

First steps in kernel analysis - Understanding the initial limiter

- GPU "Speed of Light Throughput"
 - SOL = theoretical peak
- "Breakdown" tables
 - DRAM: Cycles Active
- Tooltips
- Rules point to next steps

Connection Debu	ug <u>P</u> rofile <u>T</u> ools <u>W</u> ir	idow <u>H</u> elp									
Connect 🗇 Disco	nnect × Terminate	🕑 Profile Kernel		¢ 🖲 →:	→- →[→]		Baseline	es 💷 🕻	> Metr	ric Details	
spmv_v100_21.5_0.n	cu-rep ×										
ge: Details	▼ Result: 0 - 545 - main_41_gpu ▼ ▼ Add Baseline ▼ Apply Rules 🗐 Occupancy Calculator						Copy as Image				
	Result		Time	Cycles	Regs GPU		SM Frequency	CC Proce	ss		
Current	545 - main_41_gpu (63	3443, 1, 1)x(128, 1, 1)	7,75 msecond	10.176.310	80 0 - Tesla 1	V100-SXM2-16G	B 1,31 cycle/nsec	ond 7.0 [1955	9] spmv		•
The report contained	ns imported source files.										1
 GPU Speed Of Lig 	ht Throughput							All		- P	
igh-level overview of naximum. Breakdowr	f the throughput for comp ns show the throughput fo	ute and memory reso r each individual sub	urces of the GPL -metric of Comp	J. For each uni ute and Memo	it, the throughput i ory to clearly ident	reports the achie tify the highest c	eved percentage of ontributor. High-leve	utilization with re l overview of the	spect to the utilization	e theoretical for compute and	
nemory resources of	the GPU presented as a re	oofline chart.									
Compute (SM) Throu	ghput [%]			3,11	Duration [mseco	nd]				7,75	
Memory Throughput [%] 92			92,37	Elapsed Cycles [10.176.310				
L1/TEX Cache Throug	X Cache Throughput [%] 32,76 SM Active Cycles [cycle]							10.160.469,39			
L2 Cache Throughput	t [%]			31,70	SM Frequency [cycle/nsecond]				1,31		
 High Through Roofline Analy 	put The kernel is utilizin shifted from the m The ratio of peak f	ng greater than 80.0% ost utilized to anothe loat (fp32) to double	6 of the available r unit. Start by ar (fp64) performar	compute or n nalyzing workle nce on this de	nemory performar oads in the <u>Merr</u> vice is 2:1. The ke	nce of the device nory Workload A ernel achieved 09	e. To further improve nalysis section. 6 of this device's fp3	e performance, w 2 peak performa	ork will like nce and clo	ly need to be ose to 1% of	
High ThroughRoofline Analy	put The kernel is utilizing shifted from the momenta of peak for the ratio of peak for the formation of peak formation of peak formation of peak for the f	ng greater than 80.09 ost utilized to anothe loat (fp32) to double rmance.	6 of the available r unit. Start by ar (fp64) performar	compute or n nalyzing workle	nemory performar oads in the <u>Merr</u> vice is 2:1. The ke	nce of the device nory Workload A ernel achieved 09	e. To further improve nalysis section. 6 of this device's fp3	: performance, w 2 peak performa	ork will like nce and clo	ly need to be ose to 1% of	1
 High Through Roofline Analy 	The kernel is utilizi shifted from the m ysis The ratio of peak f its fp64 peak perfo	ng greater than 80.0% ost utilized to anothe loat (fp32) to double rmance.	6 of the available r unit. Start by ar (fp64) performar	compute or n nalyzing workle nce on this de GPU Th	nemory performar oads in the <u>▶ Merr</u> vice is 2:1. The ker roughput	nce of the device nory Workload A ernel achieved 09	e. To further improve <u>nalysis</u> section. 6 of this device's fp3	e performance, w 2 peak performa	ork will like	ly need to be ose to 1% of	
High Through Roofline Anal Compute (SM) [%]	put The kernel is utilizing shifted from the mashifted from th	ng greater than 80.03 sst utilized to anothe loat (fp32) to double rrmance.	6 of the available r unit. Start by ar (fp64) performar	compute or n nalyzing workle nce on this de GPU Th	nemory performar oads in the <u>▶ Men</u> vice is 2:1. The ke roughput	nce of the device nory Workload A ernel achieved 09	e. To further improve nalysis section. S of this device's fp3	e performance, w 2 peak performa	ork will like	ly need to be	1
 High Through Roofline Analy Compute (SM) [%] Memory [%] 	put The kernel is utilizit shifted from the main shi	ng greater than 80.0% ost utilized to anothe loat (fp32) to double rmance.	5 of the available r unit. Start by ar (fp64) performat	compute or n nalyzing workle nce on this de GPU Th	nemory performar oads in the <u>▶ Men</u> vice is 2:1. The ker roughput	nce of the device nory Workload A ernel achieved 09	t. To further improve nalysis section.	2 performance, w	ork will like	ly need to be	
 High Through Roofline Analy Compute (SM) [%] Memory [%] 0, 	put The kernel is utilizit shifted from the minimative f	ng greater than 80.03 sst utilized to anothe oat (fp32) to double rmance. 20,0	s of the available r unit. Start by ar (fp64) performan g0,0	compute or n nalyzing workle nce on this der GPU Th 40,0	nemory performar oads in the ▶ Merr vice is 2:1. The ker roughput	nce of the device nory Workload A rrnel achieved 09 60,0	b. To further improve nalysis section. a of this device's fp3 70,0	2 performance, w 2 peak performa 80,0	ork will like nce and clc 90,0	ly need to be ose to 1% of	1
High Through Roofline Analy Compute (SM) [%] Memory [%] 0,	put The kernel is utilizit shifted from the minimation of peak fits fp64 peak performed and the shifted from the minimatic shi	ng greater than 80.03 sst utilized to anothe oat (fp32) to double rmance. 20,0	s of the available r unit. Start by ar (fp64) performan g0,0	compute or n nalyzing workle nee on this der GPU Th 40,0 Spec	nemory performar oads in the <u>Merr</u> vice is 2:1. The ker roughput 50,0 eed Of Light (SOL)	nce of the device nory Workload A rrnel achieved 09 60,0) [%]	b. To further improve nalysis section. a of this device's fp3 70,0	2 performance, w 2 peak performa 80,0	ork will like nce and clo 90,0	ly need to be ose to 1% of 100,0	
High Through Roofline Anal Compute (SM) [%] Memory [%] 0,	put The kernel is utilizit shifted from the mini- ysis The ratio of peak f its fp64 peak perfor 0 10,0 Compute Thro	ng greater than 80.03 pst utilized to anothe loat (fp32) to double rmance. 20,0 pughput Breakdo	s of the available r unit. Start by ar (fp64) performan 30,0 wm	compute or n nalyzing workle nce on this der GPU Th 40,0 Spe	nemory performar oads in the ▶ Merr vice is 2:1. The ker roughput 50,0 eved Of Light (SOL)	nce of the device nory Workload A ernel achieved 09 60,0) [%] Mer	b. To further improve nalysis section. a of this device's fp3 70,0 nory Throughput	2 performance, w 2 peak performa 80,0 It Breakdown	ork will like nce and clo 90,0	ly need to be use to 1% of 100,0	
 High Through Roofline Anal Compute (SM) [%] Memory [%] 0, SM: Mio2rf Writel 	put The kernel is utilizit shifted from the minipage ysis The ratio of peak fits fp64 peak performed 0 10,0 Compute Throe back Active [%]	ng greater than 80.03 sst utilized to anothe loat (fp32) to double rmance. 20,0 pughput Breakdo	s of the available r unit. Start by ar (fp64) performan 30,0 wm	compute or n nalyzing workl nce on this der GPU Th 40,0 Spe 3,11	nemory performar bads in the Merr vice is 2:1. The ker roughput 50,0 wed Of Light (SOL)	nce of the device nory Workload A ernel achieved 09 60,0) [%] Mer Active [%]	b. To further improve nalysis section. s of this device's fp3 70,0 nory Throughpu	2 performance, w 2 peak performa 80,0 It Breakdown	ork will like nce and clo 90,0	ly need to be use to 1% of 100,0 92,37	1
 High Through Roofline Analy Compute (SM) [%] Memory [%] 0, SM: Mio2rf Writel SM: Inst Executed 	put The kernel is utilizit shifted from the minipage The ratio of peak f ysis The ratio of peak f its fp64 peak performed 10,0 0 10,0 Compute Three back Active [%] 10,0	ng greater than 80.03 pst utilized to anothe loat (fp32) to double rmance. 20,0 pughput Breakdo	s of the available r unit. Start by ar (fp64) performan 30,0	compute or n nalyzing workle nee on this der GPU Th 40,0 Spe 3,11 2,74	nemory performar oads in the <u>Merr</u> vice is 2:1. The ker roughput 50,0 eed Of Light (SOL) DRAM: Cycles DRAM: Dram S	nce of the device nory Workload A ernel achieved 09 60,0) [%] Mer : Active [%] Sect dram_cycl	 To further improve nalysis section. of this device's fp3 70,0 nory Throughpu es_active.avg.pct_o 	2 performance, w 2 peak performa 80,0 It Breakdown	ork will like noe and clo 90,0	ly need to be ase to 1% of 100,0 92,37	
 High Through Roofline Analy Compute (SM) [%] Memory [%] 0, SM: Mio2rf Writel SM: Inst Executed SM: Issue Active 	put The kernel is utilizit shifted from the minima ysis The ratio of peak f its fp64 peak performance 0 10,0 Compute Thrce back Active [%] d Pipe Lsu [%]	ng greater than 80.03 pst utilized to anothe oat (fp32) to double rmance. 20,0 pughput Breakdo	s of the available r unit. Start by ar (fp64) performan 30,0	compute or n nalyzing workle GPU Th 40,0 Spe 3,11 2,74 1,84	nemory performar bads in the <u>Merr</u> vice is 2:1. The kei roughput 50,0 eed Of Light (SOL) DRAM: Cycles DRAM: Dram S L2: D Sectors I	nce of the device nory Workload A ernel achieved 09 60,0) [%] Mer Active [%] Sect dram_cycl Fill d # of cycles	b. To further improve nalysis section. a of this device's fp3 5 of this device's fp3 70,0 nory Throughput es_active.avg.pct_o where DRAM was a	2 performance, w 2 peak performa 80,0 It Breakdown f_peak_sustained ctive	ork will like nce and clc 90,0 1 1_elapsed	ly need to be ose to 1% of 100,0 92,37	
 High Through Roofline Analy Compute (SM) [%] Memory [%] 0, SM: Mio2rf Writel SM: Inst Executed SM: Inst Executed SM: Inst Executed 	put The kernel is utilizing shifted from the main shifted from t	ng greater than 80.03 sst utilized to anothe loat (fp32) to double rmance. 20,0 pughput Breakdo	s of the available r unit. Start by ar (fp64) performan 30,0	compute or n nalyzing workle nee on this der GPU Th 40,0 Spe 3,11 2,74 1,84 1,84	nemory performar oads in the <u>Merr</u> vice is 2:1. The kei roughput 50,0 ceed Of Light (SOL) DRAM: Cycles DRAM: Cycles DRAM: Dram S L2: D Sectors I L1: Data Pipe I	60,0 60,0	b. To further improve nalysis section. a of this device's fp3 5 of this device's fp3 70,0 nory Throughput es_active.avg.pct_o where DRAM was a be (main) memory, v	2 peak performance, w 2 peak performa 80,0 It Breakdown f_peak_sustained ctive	ork will like nce and clc 90,0 1 1_elapsed lobal and lc	ly need to be ose to 1% of 100,c 92,37	des
 High Through Roofline Analy Compute (SM) [%] Memory [%] 0, SM: Mio2rf Writel SM: Inst Executed SM: Issue Active SM: Mio Inst Essue SM: Mio Inst Essue 	put The kernel is utilizit shifted from the million The ratio of peak f ysis The ratio of peak f tis fp64 peak perform The ratio of peak f 0 10,0 Compute Three The compute Three back Active [%] The compute Three [%] d [%] ued [%] The compute Three	ng greater than 80.03 sst utilized to anothe ioat (fp32) to double rrmance. 20,0 20,0	s of the available r unit. Start by ar (fp64) performar 30,0	compute or n nalyzing workl nce on this der GPU Th 40,0 Spe 3,11 2,74 1,84 1,84 1,84	nemory performar aads in the Merrivice is 2:1. The ker roughput 50,0 ceed of Light (SOL) DRAM: Cycles DRAM: Drams L2: D Sectors I L1: Data Pipe I L1: Lsu Writeb	acce of the device nory Workload A ernel achieved 09 60,0) [%] 60,0) [%] Mer Active [%] Sect dram_cycl Fill d # of cycles Eud dram_Devic pack Active [%]	2. To further improve nalysis section. 5 of this device's fp3 70,0 nory Throughpu es_active.avg.pct.o where DRAM was a be (main) memory, v	2 peak performance, w 2 peak performa 2 peak performa 80,0 It Breakdown f_peak_sustained tive	ork will like nce and clc 90,0 1 1 1 2 2 2 3 2 1 2 2 3 2 0,0	ly need to be pse to 1% of 100,0 92,37 pocal memory res 25,74	des
 High Through Roofline Analy Compute (SM) [%] Memory [%] Memory [%] SM: Mio2rf Writel SM: Inst Executed SM: Inst Executed SM: Inst Executed SM: Nio Inst Issu SM: Pipe Fp64 Cy 	put The kernel is utilizit shifted from the minima ysis The ratio of peak f its fp64 peak performance 0 10,0 Compute Three back Active [%] d Pipe Lsu [%] [%] d [%] ue [%] ycles Active [%]	ng greater than 80.03 sist utilized to anothe loat (fp32) to double irmance. 20,0 20,0 20,0 20,0	s of the available r unit. Start by ar (fp64) performan 30,0	compute or n nalyzing workl nce on this der GPU Th 40,0 Spe 3,11 2,74 1,84 1,84 1,84 1,84 0,84	nemory performar oads in the Merr vice is 2:1. The ker roughput 50,0 eed Of Light (SOL) DRAM: Cycles DRAM: Dram S L2: D Sectors I L1: Data Writeb L2: T Sectors [nce of the device nory Workload A ernel achieved 09 60,0) [%] Mer Active [%] Seet dram_cycl Fill c # of cycles Lsu dram: Devic ack Active [%]	E. To further improve nalysis section. S of this device's fp3 70,0 nory Throughpu es_active.avg.pct_o where DRAM was a se (main) memory, v	2 peak performance, w 2 peak performa 80,0 It Breakdown f_peak_sustained ctive	ork will like noe and old 90,0 1 1_elapsed lobal and lo	ly need to be ose to 1% of 100,0 92,37 92,37	des
 High Through Roofline Anal Compute (SM) [%] Memory [%] Memory [%] SM: Mio2rf Writel SM: Inst Executed SM: Pipe Fp64 Cy SM: Pipe Shared 	put The kernel is utilizit shifted from the minima ysis The ratio of peak f its fp64 peak performance 0 10,0 Compute Three back Active [%] d [%] ice[%] ycles Active [%] Cycles Active [%]	ng greater than 80.03 set utilized to anothe ioat (fp32) to double irmance. 20,0 20,0 Dughput Breakdo	s of the available r unit. Start by ar (fp64) performan 30,0	compute or n nalyzing workl nce on this der GPU Th 40,0 Spe 3,11 2,74 1,84 1,84 1,84 1,84 1,84 0,84 0,84	nemory performar bads in the ▶ Merr vice is 2:1. The ker roughput 50,0 wed Of Light (SOL) DRAM: Cycles DRAM: Dram S L2: D Sectors I L1: Data Pipe L L1: Lsu Writeb L2: T Sectors [L2: L1:S2xbar C	nce of the device nory Workload A rmel achieved 09 60,0) [%] Mer Active [%] Sect dram_cycl Fill d f of cycles Lsu dram. Devic ack Active [%] Sycles Active [%]	To further improve nalysis section. of this device's fp3 70,0 70,0 rory Throughpu es_active.avg.pct.o where DRAM was a be (main) memory, v	2 peak performance, w 2 peak performa 80,0 It Breakdown f_peak_sustained there the GPUs g	ork will like noe and old 90,0 1 1_elapsed lobal and lo	ly need to be use to 1% of 100,0 92,37 92,37	des
 High Through Roofline Anal Compute (SM) [%] Memory [%] Memory [%] SM: Mio2rf Writel SM: Inst Executed SM: Pipe Fp64 Cy SM: Pipe Fp4 Ucyci SM: Pipe Alu Cyci 	put The kernel is utilizit shifted from the minimation of peak from the minimation of the minima	ng greater than 80.03 sst utilized to anothe oat (fp32) to double rmance. 20,0 bughput Breakdo	s of the available r unit. Start by ar (fp64) performan 30,0	compute or n nalyzing workl nce on this der GPU Th 40,0 Spe 3,11 2,74 1,84 1,84 1,84 1,84 0,84 0,84 0,84	nemory performar oads in the ▶ Merr vice is 2:1. The ker roughput 50,0 wed Of Light (SOL) DRAM: Cycles DRAM: Cycles DRAM: Cycles DRAM: Dram S L2: D Sectors I L1: Data Pipe I L1: Law Writeb L2: T Sectors I L2: Lts2xbar C L2: Lts2xbar C	nce of the device nory Workload A rrnel achieved 0% 60,0) [%] Mer Active [%] Sect dram_cycl Fill (# of cycles Lsu dram: Devic ack Active [%] Sycles Active [%] Sycles Active [%]		2 peak performance, w 2 peak performa 80,0 It Breakclown f_peak_sustained tive	ork will like noe and clo 90,0 1 1_elapsed Iobal and lo	ly need to be use to 1% of 100,0 92,37 0,74 24,55 23,90 21,23 0,74 1,23 0,74 1,23 0,74 1,23 0,74 1,23 0,74 1,23 0,74 1,23 0,74 1,23 0,74 1,23 1,35 1,35 1,35 1,35 1,35 1,35 1,35 1,35 1,35 1,35 1,35 1,35 1,35 1,35 1,35 1,35 1,	des

27 nviqi_{ke} 3

KERNEL-LEVEL PROFILING

Performance limiter categories



Motivating Example: Matrix Transpose

No FLOPs



ordering

Using Nsight Compute

▼ Source Counters			Q			
Source metrics, including branch efficiency and samp indicate when warps were stalled and couldn't be sch issue every cycle.	bled warp stall reasons. Warp s reduled. See the documentation	Stall Sampling metrics are periodically sampled over the kernel runtime. The on for a description of all stall reasons. Only focus on stalls if the scheduler:	y source	: transpose.cu V GD Navigate By: Warp Stall Sampling (All Samples)	Redo Resolve	Source
anch Instructions [inst] 4194304 Branch Efficiency [%]			0	# Source	Instructions p Stall Sam	npling Address
Branch Instructions Ratio [%]	0.07 Avg. Divergent Branches			13 global word transposed Integer+ const a trans const Integer+ const a	3.33% 0.08%	ipies) Space
Uncoalesced Global Accesses total 83886 locations. T accesses.	has uncoalesced global acce 5080 sectors). Check the L2 T The ⊕ CUDA Programming Gu	sses resulting in a total of 50331648 excessive sectors (60% of the heoretical Sectors Global Excessive table for the primary source ide had additional information on reducing uncoalesced device memory	•	<pre>14 { 15 const Integer col_block = blockIdx.x; 16 const Integer row_block = blockIdx.y; 17 const Integer block_col = threadIdx.x; 18 const Integer block_row = threadIdx.y;</pre>	3.33% 0.05% 3.33% 0.06% 6.67% 2.58% 6.67% 1.91%	
	L2 Theoretical Secto	rs Global Excessive		<pre>19 const Integer col = col_block+BLOCK_SIZE+block_col;</pre>	3.33% 0.98%	
Location		Value Val	Je (%)	<pre>20 const Integer row = row_block+BLOCK_SIZE+block_row;</pre>	3.33% 0.95%	
transpose.cu:31 (0x14d5474514c0 in transpose(I 루 transpose.cu:31 (0x14d547451450 in transpose(I 루		50.331.648 0	0	21 22 //TODO: declare shared memory for tile 23 //shared a_tile 24		
Warp Stall Sampling (All Sample	s)	Most Instructions Executed		25 if (rom < n && col < n)	16.67% 0.73%	
Location Value transpose.cu:33* 167.816 transpose.cu:31* 110.516 transpose.cu:17* 7.944 transpose.cu:18* 5.506 transpose.cu:31* 4.163	Value (%) Location transpose.cu: 36 transpose.cu: 1 transpose.cu: 1 transpose.cu: 1 transpose.cu:	Value Value <th< td=""><td>ıe (%) 3) 3) 3) 3) 3) 3) ▲ ></td><td><pre>26 { 27 //TODO: load tile of a into shared memory 28 //TODO: callsyncthreads() to ensure all shared memory writes are c 29 //TODO: read from a_tile with correct index: 30 //A_trans[col_block=BLOCK_SIZE+block_rom) * n + (rom_block=BLOCK_SIZ 31 a_trans[col+n+rom] = a[rom+n+col]; </pre></td><td>50.00% 38.57%</td><td>Global(2)</td></th<>	ıe (%) 3) 3) 3) 3) 3) 3) ▲ >	<pre>26 { 27 //TODO: load tile of a into shared memory 28 //TODO: callsyncthreads() to ensure all shared memory writes are c 29 //TODO: read from a_tile with correct index: 30 //A_trans[col_block=BLOCK_SIZE+block_rom) * n + (rom_block=BLOCK_SIZ 31 a_trans[col+n+rom] = a[rom+n+col]; </pre>	50.00% 38.57%	Global(2)
			Â	32 } 33 } 34 35 int main()	3.33% 54.11%	Total Sample Count: 119637 80.18% Long Scoreboard (95921) 14.68% Lg Throttle (17563) 1.34% Wait (1599) 1.25% Not Selected (1497)
			4		4	1.21% Math Pipe Throttle (1453)
uncoalesced al	obal excess	es [] 60% of the total"	Inline	S Transformer Supersonal Source Markers		0.50% Selected (602) 0.44% Dispatch Stall (524) 0.38% Mio Throttle (454) 0.01% No Instructions (17) 0.01% Imc Miss (7)
"uncoateseed ge	obul cheess		File	Line/Address Marker		-
			- t	anspose.cu (3)		
				Uncoalesced Global Accesses 31 * 🔥 75.	00% of this line's global ac	cesses are excessive.

Global, Local, L1, L2?

Understanding the memory hierarchy



Memory Transactions and Coalescing

- Access to global memory triggers transactions (<u>Device Mem Access</u>)
- Memory access granularity = 32 bytes = 1 sector
- Cache line = 128 bytes = 4 consecutive sectors
 - Example: 4 byte per thread \rightarrow 4B * 32 threads (1 warp) = 128B
- Data goes from **global** device memory through L2 cache
- Granularity*: Sector for L2, Cache line for L1



*The full picture: <u>S32089: Understanding and</u> <u>Optimizing Memory-Bound</u> <u>Kernels with Nsight Compute</u>

Memory Transactions and Coalescing

Coalescing details

- Coalescing: Adjacent accesses can share transactions
- Transactions must be "Naturally Aligned": First address % size == 0
- All bytes in a transaction are transferred. Use them!

For example, if a 32-byte memory transaction is generated for each thread's 4byte access, throughput is divided by 8.



degree of coalescing =

#bytes requested

#bytes transferred

Accessing Global Memory

optimal access pattern (4byte words) – fully coalesced

int x_val = x[threadIdx.x];

All addresses fall within 4 sectors

Bus utilization: 100%


worst case access pattern (4byte words) - fully uncoalesced

// stride 32
int x_val = x[32*threadIdx.x];
// "random" (pointer chasing, lists, tree, ...)
int x_val = x[lookup[threadIdx.x]];

All addresses fall in 32 different sectors

Bus utilization: 12.5%



shifted access

int x_val = x[threadIdx.x+1];

All addresses fall within 5 sectors

Bus utilization: 80% = 128B/160B



common access pattern: stride 3

int x_val = x[3*threadIdx.x];

All addresses fall within 12 sectors (4 byte words)

Bus utilization: 33%



struct {float x,y,z;} a; ... a[tid].x

 \rightarrow use structure-of-arrays (SoA): a.x[tid]

float a[M][N]; ... a[tid][42]

→ multi-dimensional arrays: pay attention to coalescing (row-major, column-major?)

another "worst case" access pattern?

// same for all threads - e.g. loop index
int x_val = x[i];

Single address, single sector

Bus utilization: 12.5%



Access pattern



Using shared memory



Block row is loaded, fully **coalesced** read

Global Memory

Shared Memory

Using shared memory



Block row is loaded, fully **coalesced** read

Indexing: Location of block is reflected on the diagonal

Global Memory

Shared Memory

Using shared memory



Block row is loaded, fully **coalesced** read

Indexing: Location of block is reflected on the diagonal

Block *column* of shared is written to *row* of matrix

Transposes the block $M \rightarrow M'$

Coalesced write

Global Memory

Shared Memory

Analysis with Nsight Compute

First steps

- Add baseline for comparison
 - Recommended: Always add --import-source true if possible
- Check the "Breakdown" tables and how they change
- Look at the other sections, warp state statistics
- Tooltips on mouseover over metrics/names/...



Kernel-level Profiling

Performance limiter categories



Analysis with Nsight Compute

Iterating and comparing

- Check the Source Counters section (also on CLI)
- Links will take you to Source/SASS view

▼ Source Counters	Q					
Source metrics, including branch efficiency and sampled warp stall reasons. Warp Stall Sampling metrics are periodically sampled over the kernel runtime. They indicate when warps were stalled and couldn't be schedulers fail to issue every cycle.						
Branch Instructions [inst] 4194304	Branch Efficiency [%] 0					
Branch Instructions Ratio [%] 0.07	Avg. Divergent Branches 0					
This kernel has uncoalesced global accesses resulting in a total of 50331648 excessive sectors (60% of the total 83886080 sectors).						

✓ Uncoalesced Global Accesses
Check the L2 Theoretical Sectors Global Excessive table for the primary source locations. The ⊕ CUDA Programming Guide had additional information on reducing uncoalesced device memory accesses.

L2 Theoretical Sectors Global Excessive

Location			Value	Value Value			
transpose.cu:31 (0x14d5474514	c0 in transpose(long long 🖉		50.331.648	8 10			
transpose.cu:31 (0x14d5474514	50 in transpose(long long 🖉		0	0			
Warp Stall Sampling (All Samples) Most Instructions Executed							
Location	Value	Value (%)	Location	Value	Value (%)		
transpose.cu:33 (0x14d5 🖉	167.816	54	transpose.cu:33 (0x14d54 🖉	2.097.152	3)		
transpose.cu:31 (0x14d5 🖉	110.516	36	transpose.cu:31 (0x14d54 🖉	2.097.152	3)		
transpose.cu:17 (0x14d5 🖉	7.948	3)	transpose.cu:31 (0x14d54 🖉	2.097.152	3)		
transpose.cu:18 (0x14d5 🖉	5.505	2	transpose.cu:31 (0x14d54 🖉	2.097.152	3)		
transpose.cu:31 (0x14d5 🖉	4.163	1	transpose.cu:31 (0x14d54 🖉	2.097.152	3)		

Roofline Analysis

How well is the hardware utilized?

Transpose does zero floating point computations... more interesting example (here: on V100)

Counting flops and transferred bytes \rightarrow AI, x-axis

Measuring achieved performance \rightarrow FLOP/s, y-axis

Rooflines from device peak bandwidth / compute



GTC session:

S32062: Performance Tuning CUDA Applications with the Roofline Model

Roofline Hackathon: https://www.youtube.com/watch?v=ZXZ2SrM3pmE&t=2382s

Branch Divergence

Recap: Warp execution

- GPUs use the Single Instruction Multiple Threads (SIMT) execution
 - functionally transparent to the programmer
 - but has performance implications

warp

- group of synchronously* executing threads (*since Volta: <u>Independent Thread Scheduling</u>)
- neighbor threads (mostly x dimension)
- basic unit of scheduling



Branch Divergence

Within Warp



divergence within warp → performance penalty
if(threadIdx.x % 2 == 0) ...

Branch Divergence

Between Warps



Conclusion

To achieve coalesced global memory access:

Usually: Fix your access pattern

Try to use shared memory (but first, check cache behavior)

Look for different way of storage or better algorithm

Avoid divergent branches

Use the tools!

GLOBAL, LOCAL, L1, L2?

Understanding the memory hierarchy



📀 NVIDIA.

65

MEMORY TRANSACTIONS AND COALESCING

Access to global memory triggers transactions (Device Mem Access)

Memory access granularity = 32 bytes = 1 sector

Cache line = **128 bytes = 4** consecutive sectors

Example: 4 byte per thread \rightarrow 4B * 32 threads (1 warp) = 128B

Here: 8 byte datatype

#bytes requested

degree of coalescing = #bytes transferred



The full picture: <u>S32089: Understanding and</u> <u>Optimizing Memory-Bound</u> <u>Kernels with Nsight Compute</u>

ANALYSIS WITH NSIGHT COMPUTE

Iterating and comparing

Add baseline for comparison

Check the "Breakdown" tables and how they change

Look at the other sections, warp state statistics

Tooltips on mouseover over metrics/names/...

Page: Details 💌 Launch: 0 - 118 - tran	spose	- Add	Baseline 🗩 Apply <u>R</u> ules]			
Current 118 Time: 1,05 msecond	Cycles: 1.149.322	Regs: 20 GPU: A100-SX	M4-40GB SM Frequency	1,09 cycle/nsecond C			
Baseline 1 118 Time: 2,66 msecond	Cycles: 2.898.470	Regs: 16 GPU: A100-SX	M4-40GB SM Frequency	1,09 cycle/nsecond C			
The report contains imported source files.							
▼ GPU Speed Of Light 🔔							
High-level overview of the utilization for compute theoretical maximum. High-level overview of the theoretical maximum.	te and memory resou e utilization for comp	urces of the GPU. For each u oute and memory resources (nit, the Speed Of Light (SO of the GPU presented as a)L) reports the achieved p roofline chart.			
SOL SM [%]		15,63 (+198,56%)	Duration [msecond]				
SOL Memory [%]		68,57 (+14,92%)	Elapsed Cycles [cycl	e]			
SOL L1/TEX Cache [%]		68,76 (+57,10%)	SM Active Cycles [cy	cle]			
SOL L2 Cache [%]		51,27 (-15,91%)	SM Frequency [cycle/	nsecond]			
SOL DRAM [%]		64,89 (+145,21%)	DRAM Frequency [cycl	e/nsecond]			
	GPU Utilization						
CM [9/]							
SIM [70]							
Memory [%]							
0,0 10,0	20,0 30	0,0 40,0	50,0 60,0	70,0			
	Speed Of Light [%]						

ANALYSIS WITH NSIGHT COMPUTE

Iterating and comparing

Branch Instructions [inst]	4.194.304 Branch Efficiency [%]	
Branch Instructions Ratio [%]	0,07 Avg. Divergent Branches	
	Sampling Data (All)	
Location	Value	Value
transpose.cu:33 (0x14beeadb03e0 in transpose) #	163.458	
transpose.cu:15 (0x14beeadb03c0 in transpose) ₽	111.189	
transpose.cu:17 (0x14beeadb0210 in transpose) ₹	7.684	
transpose.cu:18 (0x14beeadb0240 in transpose) ₽	5.406	
transpose.cu:15 (0x14beeadb0350 in transpose) ₹	3.43	
	Sampling Data (Not Issued)	
Location	Value	Value
transpose.cu:33 (0x14beeadb03e0 in transpose) #	159.279	
transpose.cu:15 (0x14beeadb03c0 in transpose) #	105.609	
transpose.cu:17 (0x14beeadb0210 in transpose) #	7.388	
transpose.cu:18 (0x14beeadb0240 in transpose) #	4.984	
transpose.cu:15 (0x14beeadb0350 in transpose) ₽	2.97	
	Most Instructions Executed	
Location	Value	Value
transpose.cu:15 (0x14beeadb0200 in transpose) ₹	2.097.152	
transpose.cu:17 (0x14beeadb0210 in transpose) #	2.097.152	
transpose.cu:17 (0x14beeadb0220 in transpose) ₽	2.097.152	
transpose.cu:15 (0x14beeadb0230 in transpose) 🖡	2.097.152	
transpose.cu:18 (0x14beeadb0240 in transpose) =	2.097.152	
	Recommendations	

Check the Source Counters section (also on CLI)

Links will take you to Source/SASS view

ANALYSIS WITH NSIGHT COMPUTE

Iterating and comparing

Page: Source 💌 Launch: 0 - 118 - transpose 💌	Add Baseline - Apply Ru	les						[Copy as Image 🔻
Current 118 - transpose (256, 256, 1)x(32, 32, 1) Time: 2,66	msecond Cycles: 2.898.470 Regs:	16 GPU: A100-SXM4-40G	B SM Frequen	cy: 1,09 cycle/nsecond CC: 8.0 Proc	ess: [7523] transpose				
View: Source and SASS 🔻									
Source: transpose.cu 💌 Find 🗸 🔨 Navigation: L2 Sect	ors Global 🗸	~ ^ 쥿 怞 묘	<u></u>	Source: transpose 💌 Find	Navigation: Instruction	s Executed	- · ^	· 18 18 주	
	Live Samp	ing Sampling Data	Instruct	<u> </u>		Live	Sampling	Sampling Data	Instruct *
# Source	Registers Data (All) (Not Issued)	Exec	# Address Sou	rce	Registers	Data (AII)	(Not Issued)	Exec
13global void transpose(Integer* const a_trans, const I		0 0		11 000014be eadb02a0	ISETP.GE.AND.EX P1, PT, R3, c[0x0]	5	216	27 🔄	2.097
14 {		0 0		12 000014be eadb02b0	ISETP.LT.U32.AND P0, PT, R4, c[0x0	5	541	95 🔄	2.097
<pre>15 const Integer col_block = blockIdx.x;</pre>	4	438 208	4.194	13 000014be eadb02c0	ISETP.LT.AND.EX P0, PT, R5, c[0x0]	5	550	65 🔄	2.097
<pre>16 const Integer row_block = blockIdx.y;</pre>	6	187 42	2.097	14 000014be eadb02d0 @!P0	9 EXIT	5	792	111 🔄	2.097
<pre>17 const Integer block_col = threadIdx.x;</pre>	7.	719 7.388	4.194	15 000014be eadb02e0	IMAD R9, R5, c[0x0][0x170], RZ	6	306	25 🔄	2.097
<pre>18 const Integer block_row = threadIdx.y;</pre>	6 5.	72 0 5.034	4.194	16 000014be eadb02f0	ULDC.64 UR4, c[0x0][0x118]	6	33	0	2.097
<pre>19 const Integer col = col_block*BLOCK_SIZE+block_col;</pre>	6 2.	305 1.691	2.097	17 000014be eadb0300	IMAD.WIDE.U32 R6, R4, c[0x0][0x170	8	50	0	2.097
<pre>20 const Integer row = row_block*BLOCK_SIZE+block_row;</pre>	6 2.	980 2.330	2.097	18 000014be eadb0310	IMAD R9, R4, c[0x0][0x174], R9	8	635	35 📃	2.097
21		0 0		19 000014be eadb0320	LEA R8, P0, R6, c[0x0][0x168], 0x3	9	660	52 📃	2.097
22 //TODO: declare shared memory for tile		0 0	-	20 000014be eadb0330	IADD3 R7, R7, R9, RZ	9	124	8 📃	2.097
23 //shared a_tile		0 0		21 000014be eadb0340	LEA.HI.X R9, R6, c[0x0][0x16c], R	9	807	75 📃	2.097
24		0 0		22 000014be eadb0350	LDG.E.64 R6, [R8.64]	9	3.437	2.977	2.097
25 if (row < n && col < n)	6 2.	342 333	10.485	23 000014be eadb0360	IMAD R3, R3, c[0x0][0x170], RZ	7	49	0	2.097
26 {		0 0		24 000014be eadb0370	IMAD.WIDE.U32 R4, R2, c[0x0][0x170	7	214	20 📃	2.097
27 //TODO: load tile of a into shared memory		0 0		25 000014be eadb0380	IMAD R3, R2, c[0x0][0x174], R3	7	887	88	2.097
28 //TODO: callsyncthreads() to ensure all shared (0 0		26 000014be eadb0390	LEA R2, P0, R4, c[0x0][0x160], 0x3	7	515	71 📃	2.097
29 //TODO: read from a_tile with correct index:		0 0		27 000014be eadb03a0	IMAD.IADD R3, R5, 0x1, R3	7	43	0	2.097
<pre>30 //a_trans[(col_block*BLOCK_SIZE+block_row) * n + (</pre>		0 0		28 000014be eadb03b0	LEA.HI.X R3, R4, c[0x0][0x164], R3	6	301	53	2.097
<pre>31 a_trans[col*n+row] = a[row*n+col];</pre>	9 119.	250 109.013	31.457	29 000014be eadb03c0	STG.E.64 [R2.64], R6	5	111.189	105.609	2.097
32 }		0 0		30 000014be eadb03d0	EXIT	1	34	0	2.097
33 }	1 163.	492 159.279	2.097	31 000014be eadb03e0	BRA 0x14beeadb03e0	0	163.458	159.279	
34		0 0		32 000014be eadb03f0	NOP		0	0	

ROOFLINE ANALYSIS

How well is the hardware utilized?

Transpose does zero floating point computations... more interesting example (here: on V100)

Counting flops and transferred bytes \rightarrow AI, x-axis

Measuring achieved performance \rightarrow FLOP/s, y-axis

Rooflines from device peak bandwidth / compute



GTC session: <u>S32062: Performance Tuning CUDA Applications with the Roofline Model</u>

Roofline Hackathon: https://www.youtube.com/watch?v=ZXZ2SrM3pmE&t=2382s

MORE DETAILS

Blocks of threads, warps

- Single Instruction Multiple Threads (SIMT) model
- CUDA hierarchy: Grid -> Blocks -> Threads
- One warp = 32 threads.
- Why does it matter ? Many optimizations based on behavior at the warp level

Mapping threads

- Thread blocks can be 1D, 2D, 3D Only for convenience. HW "looks" at threads in 1D
- **Consecutive 32 threads** belong to the same warp



Mapping threads

- Thread blocks can be 1D, 2D, 3D
 Only for convenience. HW "looks" at threads in 1D
- **Consecutive 32 threads** belong to the same warp



Control Flow

- Different warps can execute different code
 No impact on performance
 Each warp maintains its own Program Counter
- Different code path inside the same warp ? Threads that don't participate are masked out, but the whole warp executes both sides of the branch



Instructions, time

A; Warp 1 if(threadIdx.y==0) B; else Warp 2 C; D; Warp 3

31

0

•••

31

0

... 31

0

•••



Instructions, time

A; Wa if(threadIdx.y==0) B; else Wa C; D; Wa





Instructions, time

A; if(threadIdx.y==0) B; else C; D;

Warp 1 $0 \\ \dots \\ 31$ ABWarp 2 $0 \\ \dots \\ 31$ $0 \\ \dots \\ 31$ Warp 3 $0 \\ \dots \\ 31$



Instructions, time

A; if(threadIdx.y==0) B; else C; D;



Instructions, time

A; if(threadIdx.y==0) B; else C; D;





Instructions, time

A; if(threadIdx.y==0) B; else C; D;





Instructions, time

A; if(threadIdx.y==0) B; else C; D;

0 Warp 1 В D A ••• 31 0 Warp 2 A В C ••• 31 0 Warp 3 ••• 31



Instructions, time

A; if(threadIdx.y==0) B; else C; D;

0 Warp 1 В D A ••• 31 0 Warp 2 A В С D ••• 31 0 Warp 3 ••• 31



Instructions, time

A; if(threadIdx.y==0) B; else C; D;


CONTROL FLOW



Instructions, time

A; if(threadIdx.y==0) B; else C; D;



CONTROL FLOW



Instructions, time

A; if(threadIdx.y==0) B; else C; D;



CONTROL FLOW

Takeaways

- Minimize thread divergence inside a warp
- Divergence between warps is fine
- Maximize "useful" cycles for each thread

HIDING LATENCY

LATENCY

GPUs cover latencies by having a lot of work in flight



The warp waits (latency)



No warp issues

SM RESOURCES

Each thread block needs:

Registers (#registers/thread x #threads) Shared memory (0 ~ 96 KB)

Volta limits per SM:

256KB Registers 96KB Shared memory 2048 threads max (64 warps) 32 thread blocks max

Can schedule any resident warp without context switch

L1 Instruction Cache																			
							L1 Instru	ctio	n Cache										
						_	_	٦Г		_					_				
L0 Instruction Cache Warn Scheduler (32 thread/clk)									L0 Instruction Cache										
Dispatch Unit (32 thread/clk)									Dispatch Unit (32 thread/clk)										
	Register File (16,384 x 32-bit)								Register File (16,384 x 32-bit)										
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32						
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32						
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32						
FP64	INT	INT	FP32	FP32	TEN	SOR	TENSOR		FP64	INT	INT	FP32	FP32	TEN	ISOR	TENSOR			
FP64	INT	INT	FP32	FP32	co	RE	CORE		FP64	INT	INT	FP32	FP32	C	JRE	CORE			
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32						
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32						
FP64	INT	INT	FP32	FP32					FP64	INT	INT	FP32	FP32						
LD/ LD/ ST ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU		LD/ LD/ ST ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	LD/ ST	SFU			
	L0 Instruction Cache										L0 Instruction Cache								
		L0 li	nstruc	tion C	ache][L0 h	nstruc	tion C	ache					
	Wa	L0 li rp Sch	nstruc 1edule	tion C r (32 t	ache hread/	clk)				War	L0 li rp Sch	nstruc 1edule	tion C r (32 t	ache hreac	l/clk)				
	Wa Di	L0 In rp Sch spatcl	nstruc nedule h Unit	tion C r (32 t (32 th	ache hread/ read/c	cik) ik)				Wai Di	LO II rp Sch spatcl	nstruc Iedule h Unit	tion C r (32 t (32 th	ache hreac iread/	i/cik) cik)				
	War Di Reg	L0 In rp Sch spatcl jister	nstruc hedule h Unit File ('	tion C r (32 t (32 th 16,384	ache hread/ read/c 4 x 32	cik) ik) -bit)				War Di Reg	L0 II rp Sch spatc lister	nstruc Iedule h Unit File (1	tion C r (32 t (32 th 16,38	ache hread read/ 4 x 3	l/clk) clk) 2-bit)				
FP64	Wai Di Reg	LO In rp Sch spatcl jister INT	nstruc nedule h Unit File (' FP32	tion C r (32 t (32 th 16,384 FP32	ache hread/ read/c 4 x 32	cik) ik) -bit)			FP64	Wai Di Reg	LO II rp Sch spatc jister INT	nstruc nedule h Unit File (* FP32	tion C r (32 t (32 th 16,38 FP32	ache hread read/ 4 x 3	l/clk) clk) 2-bit)				
FP64 FP64	Wai Di Reg INT	L0 In rp Sch spatcl jister INT INT	nstruc hedule h Unit File (' FP32 FP32	tion C r (32 t (32 th 16,38 FP32 FP32	ache hread/ read/c 4 x 32	cik) ik) -bit)			FP64 FP64	War Di Reg INT	LO II rp Sch spatc jister INT INT	nstruct hedule h Unit File (* FP32 FP32	tion C r (32 th (32 th 16,38 FP32 FP32	ache hread/ read/ 4 x 3	l/clk) clk) 2-bit)				
FP64 FP64 FP64	War Di Reg INT INT	L0 In p Sch spatcl ister INT INT INT	nstruc redule h Unit File (' FP32 FP32 FP32	tion C (32 t (32 th 16,384 FP32 FP32 FP32	ache hread/ read/c 4 x 32	clk) Ik) -bit)			FP64 FP64	War Dis Reg INT INT	L0 II rp Sch spatc jister INT INT INT	nstruc hedule h Unit File (* FP32 FP32 FP32	tion C r (32 th (32 th 16,38 FP32 FP32 FP32	ache hread/ read/	i/cik) cik) 2-bit)				
FP64 FP64 FP64 FP64	Wai Di Reg INT INT INT	L0 In rp Sch spatcl ister INT INT INT	nstruc hedule h Unit File (* FP32 FP32 FP32	tion C (32 t (32 th 16,384 FP32 FP32 FP32 FP32 FP32	ache hread/ read/c 4 x 32	cik) ik) -bit) SOR	TENSOR		FP64 FP64 FP64 FP64	War Di Reg INT INT INT	L0 II rp Sch spatc jister INT INT INT	nstruc hedule h Unit File (* FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,38 FP32 FP32 FP32 FP32	ache hread read/ 4 x 3:	I/clk) clk) 2-bit)	TENSOR			
FP64 FP64 FP64 FP64 FP64	War Di Reg INT INT INT INT	L0 In p Sch jister INT INT INT INT	nstruc hedule h Unit File (' FP32 FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,384 FP32 FP32 FP32 FP32 FP32	ache hread/ read/c 4 x 32 TEN CO	cik) ik) -bit) SOR RE	TENSOR		FP64 FP64 FP64 FP64 FP64	Wai Dis Reg INT INT INT INT	L0 In rp Sch spatc ister INT INT INT INT	nstruct nedule h Unit File (* FP32 FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,38 FP32 FP32 FP32 FP32 FP32	iache hreac iread/ 4 x 3	I/clk) clk) 2-bit) 2-bit 2-bit	TENSOR			
FP64 FP64 FP64 FP64 FP64 FP64	War Di Reg INT INT INT INT	LO In rp Sch spatcl spatcl ister INT INT INT INT INT	nstruc nedule h Unit File (FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,384 FP32 FP32 FP32 FP32 FP32 FP32	ache hread/ read/c 4 x 32	cik) ik) -bit) SOR RE	TENSOR		FP64 FP64 FP64 FP64 FP64 FP64	War Di: Reg INT INT INT INT	LO II rp Sch spatc spatc ister INT INT INT INT INT	FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,38 FP32 FP32 FP32 FP32 FP32 FP32	tache hread/ 4 x 3. TEM	i/clk) clk) 2-bit) NSOR DRE	TENSOR			
FP64 FP64 FP64 FP64 FP64 FP64 FP64	Wan Di Reg INT INT INT INT INT	LO II rp Sch spatcl ister INT INT INT INT INT INT	nstruc nedule h Unit File (' FP32 FP32 FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,38- FP32 FP32 FP32 FP32 FP32 FP32 FP32	ache hread/ read/c 4 x 32	cik) ik) -bit) SOR RE	TENSOR		FP64 FP64 FP64 FP64 FP64 FP64 FP64	Wan Di: Reg INT INT INT INT INT	LO II rp Sch spatc jister INT INT INT INT INT INT	nstruc nedule h Unit File (* FP32 FP32 FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,38 FP32 FP32 FP32 FP32 FP32 FP32 FP32	iache hread/ 4 x 3: TEN C(i/clk) clk) 2-bit) NSOR DRE	TENSOR			
FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Waa Di Regg INT INT INT INT INT INT	LO II pp Sch spatcl ister INT INT INT INT INT INT INT INT	nstruc nedule h Unit File (FP32 FP32 FP32 FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,384 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	ache hread/ 4 x 32 TEN CO	cik) ik) -bit) SOR RE	TENSOR		FP64 FP64 FP64 FP64 FP64 FP64 FP64	Wan Di: Regg INT INT INT INT INT INT	LO II rp Sch spatc spatc ister INT INT INT INT INT INT INT	nstruci nedulei h Unit File (* FP32 FP32 FP32 FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,38 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	TEN C	I/clk) clk) 2-bit) NSOR DRE	TENSOR			
FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Waa Di Regg INT INT INT INT INT INT INT	LO II p Sch spatcl ister INT INT INT INT INT INT INT INT INT	nstruc nedule h Unit File (FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	tion C (32 t (32 th 16,384 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	ache hread/ 4 x 32 TEN CO	cik) ik) -bit) SOR RE	TENSOR CORE		FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Wan Di Reg INT INT INT INT INT INT INT INT LD/ ST	LO II rp Sch spatcl ister INT INT INT INT INT INT INT INT	nstruc nedule h Unit File (7 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,38 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	Heache Hread/ 4 x 3: TEM C((/clk) clk) 2-bit) 2-bit) NSOR DRE	TENSOR CORE			
FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	Wan Di Reg INT INT INT INT INT INT INT INT	LO In p Sch spatcl ister INT INT INT INT INT INT INT INT INT	nstruc nedule h Unit File (° FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,384 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	ache hread/c 4 x 32 TEN: CO	cik) ik) -bit) sor RE	TENSOR CORE SFU	che	FP64 FP64 FP64 FP64 FP64 FP64 FP64 FP64	War Di Regg INT INT INT INT INT INT INT INT INT INT	L0 In p Sch spatcl ister INT INT INT INT INT INT INT INT	nstrucci nedule h Unit File (* FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	tion C (32 th (32 th 16,38 FP32 FP32 FP32 FP32 FP32 FP32 FP32 FP32	LD/ ST	I/clk) clk) 2-bit) VSOR DRE	TENSOR CORE			

OCCUPANCY

Achieved number of threads per SM

Maximum number of threads per SM

(Use the occupancy calculator XLS in CUDA Toolkit)

Higher occupancy can help to hide latency!

Occupancy =

SM has more warp candidates to schedule while other warps are waiting for instructions to complete

Achieved occupancy vs theoretical occupancy Need to run enough thread blocks to fill all the SMs!

LATENCY AT HIGH OCCUPANCY

Many active warps but with high latency instructions



INCREASING IN-FLIGHT INSTRUCTIONS

Ways to improve parallelism:

- Expose enough parallelism have O(10 x number of CUDA cores) threads
- Improve occupancy
 More threads -> more instructions
- Improve instruction parallelism (ILP)
 More independent instructions per thread

ADDITIONAL REFERENCES

GTC '21 talk from Nsys team: <u>https://www.nvidia.com/en-us/on-demand/session/gtcspring21-s31566</u>

Self-paced lab GTC lab by the Nsight team: <u>https://github.com/NVIDIA/nsight-training</u>

https://developer.nvidia.com/nsight-systems and https://developer.nvidia.com/nsight-compute + usage tips, videos on these pages

More explanation on "Long Scoreboard Stall" and other warp states: <u>https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html#statistical-sampler</u>

Nsight Compute is heavily customizable via Sections/Rules: <u>https://docs.nvidia.com/nsight-compute/CustomizationGuide/index.html</u>

For really advanced users: <u>https://docs.nvidia.com/nsight-compute/CustomizationGuide/index.html#report-file-format</u>

