OPTIMIZING APPLICATION PERFORMANCE WITH CUDA PROFILING TOOLS

Mayank Jain, 11 May 2017



AGENDA

- CUDA Profiling Tools
- Unified Memory Profiling
- NVLink Profiling
- PC Sampling
- MPI Profiling
- Multi-hop Remote Profiling
- Volta Support
- Other Improvements

CUDA PROFILING TOOLS

- NVIDIA® Visual Profiler 👌 🚝 📹
- nvprof 👌 📢 💣 📫 *
- NVIDIA® Nsight[™] Visual Studio Edition

* Android CUDA APK profiling not supported (yet)



3RD PARTY PROFILING TOOLS



TAU Performance System ®



VampirTrace





PAPI CUDA Component

HPC Toolkit



UNIFIED MEMORY PROFILING

UNIFIED MEMORY EVENTS



SEGMENT MODE TIMELINE OPTIONS

😣 🗈 Create New Session
Profiling Options
Set the profiling options
Profiling Options Timeline Options
Execution timeout: Enter maximum execution timeout in seconds [optional] seconds
Start execution with profiling enabled
Enable concurrent kernel profiling
Senable CUDA API tracing in the timeline
Enable power, clock, and thermal profiling
S Enable unified memory profiling
Solution width segments for Unified memory timeline
Number of segments Specify the number of segments for unified memory timelines [default 100]
Track memory allocations
Replay application to collect events and metrics [not supported with multiprocess profiling]
Profile execution on the CPU
Senable OpenACC profiling
□ Enable CPU thread tracing Uption to specify number of segments
🞯 Run guided analysis
Advanced
< Back Next > Cancel Finish



SEGMENT MODE TIMELINE

🕻 *NewSession1 🛙															- 1	■ Properties 🛱					
Process "iarohi" (5267)		0.2 s	0.25 s	0.3 s	0.35 s	0.4 s	0.45 s	0.5 s	0.55 s	0.6 s	0.65 s	0.7 s	0.75 s	0.8 s	0.85 s	CPU Page Faults					
Thread 655341440 Runtime API Driver API	-															The segment mode is used for this timeline. In this mode the timeline into equal width segments and only aggregated data values for each segment are shown.					
- Profiling Overhead																Timestamp		268.60478 ms (268,60	4,778 ns)		
Unified Memory			_													End		278.93573 ms (278,93	5,731 ns)		
CPU Page Faults															_	Duration		10.33095 ms (10,330,9	953 ns)		
 [0] Graphics Device Unified Momony 			4													Virtual Address F	Rande	0x90000000 - 0x9001	IFF000		
- T Data Migration	DtoH)															Process	9-	5267			
└ 𝕎 GPU Page Faults																110ccbb		5201			
- 🍸 Data Migration	HtoD)																				
Context 1 (CUDA)																					
Compute																	h	to the set			
└ ▼ 100.0% jacob	i_iter															The number of CPU	page faults per seco	ond within the segment			
Streams																0 - 10 %	0 - 30000]				
- Derault																10-20% [30000 - 60000]				
																20-30% [60000 - 90000]				
																30 - 40 % [90000 - 120000]				
	Coo				1t			H	eat	man	for	CPU				40 - 50 % [120000 - 150000]				
	Sea	gmei	nt m	loae	INTE	erval										50 - 60 % [150000 - 180000]				
									pa	ige t	aults	S				60 - 70 % [180000 - 210000]				
																70-80% [210000 - 240000]				
																80 - 90 % [240000 - 270000]				
																90 - 100 %	> 270000]				
																	-				

SWITCH TO NON-SEGMENT VIEW

U	ncheck Select settings view	
🗔 Analysis 🛅 GPU	U Details (Summary) 🌐 CPU Details 📩 OpenACC Details 📮 Console 🗔 Settings 🛱	
Session NewSe	ssion1	
Executable	Jse fixed width segments for Unified memory timeline	
Timeline Options	Number of segments 100	
Analysis		
	Load data for time range	
	Start time 225 ms to End time 700	ms
	Enable timelines in session view	
	▼ All	
	V V Process	
Select this t	 Profiling Overhead Markers and Ranges OpenACC Pthread CPU Page Faults Device Unified Memory Context 	
	Open in new session Apply	

NON-SEGMENTED MODE TIMELINE



🕑 Load da	ta for time range			
Start time	225	ms to End time	700	ms

CPU PAGE FAULT SOURCE CORRELATION

💺 *NewSession1 🛛 💺 *N	ewSession1-clone 🛙				_				
	0.25 s	0.3 s	0.35 s	Selected int	erval 0.5 s	Source	location	0.65 s	0.7 s
Process "jacobi" (5267)									
Thread 655341440									
Runtime API									
Driver API									
Profiling Overhead									
Unified Memory									
- 🍸 CPU Page Faults									
[0] Graphics Device									
Unified Memory									
🗏 🍸 Data Migration (Dte	oH)								
🗆 🍸 GPU Page Faults									
- 🍸 Data Migration (Hto	DD)								
Context 1 (CUDA)									
Compute									
⊢ 🍸 100.0% jacobi_it	er								
Streams									
L Default									

Properties X	
CPU Page Faults	
Timestamp	440.45958 ms (4 <mark>4</mark> 0,459,581 ns)
Memory Acccess Type	Write
Virtual Address	0x900100000
Source Location	main@jacobi.cu:130
Process	25684

11 📀 nvidia

CPU PAGE FAULT SOURCE CORRELATION

Properties X	
CPU Page Faults	
Timestamp	440.45958 ms (440,459,581 ns)
Memory Acccess Type	Write
Virtual Address	0x900100000
Source Location	main@jacobi.cu:130
Process	25684
	1

Source line causing CPU page fault

```
💺 *NewSession1  🗟 iacobi.cu 🖾
      float * a:
      float * a new:
      float * weights;
      CUDA CALL(cudaMallocManaged(&a,
                                          nx*nv*sizeof(float))):
      CUDA CALL(cudaMallocManaged(&a new, nx*ny*sizeof(float)));
      CUDA CALL(cudaMallocManaged(&weights, n weights*sizeof(float)));
      init(a,a new,nx,ny,weights,n weights);
      cudaEvent t start,stop;
      CUDA CALL(cudaEventCreate(&start));
      CUDA CALL(cudaEventCreate(&stop));
      CUDA CALL(cudaDeviceSynchronize());
      CUDA CALL(cudaEventRecord(start));
      PUSH RANGE("while loop",0)
      int iter = 0:
      while ( iter <= iter max )</pre>
          PUSH RANGE("jacobi step",1)
          jacobi iteration<<<dim3(nx/32,ny/4),dim3(32,4)>>>(a new,a,nx,ny,weights[0]);
          CUDA CALL(cudaGetLastError());
          CUDA CALL(cudaDeviceSynchronize());
          POP RANGE
          std::swap(a,a new);
          PUSH RANGE("periodic boundary conditions",2)
          //Apply periodic boundary conditions
          for (int ix = 0; ix < nx; ++ix)
                  0*nx+ix]=a[(ny-2)*nx+ix];
             a
              a[(nv-1)*nx+ix]=a[ 1*nx+ix]:
          POP RANGE
          if ( 0 == iter%100 )
              std::cout<<iter<<std::endl;</pre>
          iter++;
      3
      CUDA CALL(cudaEventRecord(stop));
      CUDA CALL(cudaDeviceSynchronize());
      POP RANGE
```

CPU PAGE FAULT SOURCE CORRELATION

Unguided Analysis		Summary of all CPU page faults							
🗏 📃 🗘 🖪 Reset All	Results								
To enable kernel analysis stages select a host-launched kernel instance in the timeline.									
Application	The following table shows the top locations where CPU page faults occurred (Double-click to open the location in source code)								
Data Movement And Concurrency 📀	CPU page faults	Source location N							
Compute Utilization	1001	main@jacobi.cu:130							
	1001	main@jacobi.cu:130							
Kernel Performance	4	Unknown							
Desendes av Aselvais	2	Unknown							
	1	_Z4initPfS_iiS_i@jacobi.cu:85							
NVLink	1	Unknown							
Unified Memory 🗸									

Option to collect Unified Memory information

NEW UNIFIED MEMORY EVENTS

Page throttling, Memory thrashing, Remote map

*unified-memory-new-events.nvvp	20													- 8
	0.312 s	0.313 s	0.314 s	0.315 s	0.316 s	0.317 s	0.318 s	0.319 s	0.32 s	0.321 s	0.322 s	0.323 s	0.324 s	0.3
Process "systemWideAtomics														
Thread 3922552640														
L Runtime API														
L Driver API														
Profiling Overhead														
Unified Memory														
- 🍸 CPU Page Faults														
🖃 🍸 Thrashing-Throttling				Page T	hrottling	Page Throttling		age Throttl		Page Throttling		Page Throttling	Page T	hrottling
- 🍸 Memory Thrashing														
- 🍸 Page Throttling		1		Page T	hrottling	Page Throttling	F	age Throttl		Page Throttling		Page Throttling	Page 1	hrottling
- 🍸 Remote Map				↑										
[0] Graphics Device														
 Unified Memory 														
- 🍸 Data Migration (DtoH)														
- 🍸 GPU Page Faults		┶┶╱┲┼┹┲												
🗆 🍸 Data Migration (HtoD)														
🖃 🍸 Thrashing-Throttling														
🗆 🍸 Memory Thrashing														
🗆 🍸 Page Throttling		/												
- 🍸 Remote Map							Q							
Context 1 (CUDA)														
Compute				_			atomicKetnel(ir	t*)						
- 🍸 100.0% atomicKer				_			atomicKernel(in	lt*)						
Streams														
L Default				_			atomicKernel(ir	t <u>I</u>						
	emory ashing			Page throt <u>tl</u> i	ng		Rem	ote ma	ap				1	1

FILTER AND ANALYZE



Select unified memory in the unguided analysis section

🔤 Analysis 🖾	🎫 GPU Details (Summary) 🖽 CPU Details 📄 OpenACC Details 📮										
	🗈 Reset All 🛛 🛄 Analyze All										
To enable kerr timeline.	nel analysis stages select a host-launched kernel instance in the										
Application											
Data Movement And Concurrency 🤣											
Compute Utilization											
Kernel Perfo	ormance 🔍 🖉										
Dependency	y Analysis 🔍 🔍										
NVLink	II , 📀										
Unified Mem	nory 🥩										



Select required events and click on 'Filter and Analyze'

Start Address: Virtual address rang	ge size:	0x900000000 End Addr 0x141000	ress: 0x900141000		
CPU Page Faults	Read	🗌 🗌 Write			
GPU Page Faults Access Type:	Read	🗌 Write	Atomic P	refetch	
HtoD Migrations Reason:	User	✓ Coherence	✓ Prefetch	Sumr	nary of
D toH Migrations	5				
Reason:	🗌 User	🗹 Coherence	🗹 Prefetch 🗌 Ev	viction	
Filter and Analyze					
he following table sl	hows the	summary of unified men	nory migrations and pa	ge faults after filtering	Ţ
Total HtoD migratio	n size To	tal DtoH migration size	Total CPU Page faults	Total GPU Page faults	Total different pages
			0	•	(0)

FILTER AND ANALYZE

Unfiltered

	233.5 ms	234 ms	234.5 ms	235 ms	235.5 ms	236 ms	236.5 ms	237 ms	237.5 ms	238 ms	238.5 ms	239 ms	239.5 ms	240 ms	240.5 ms	241 ms
Process "vecAdd_managed" (
Thread 3890149184																
Runtime API													cu	daDeviceSynchr	onize	
L Driver API																
Profiling Overhead																
Unified Memory																
🗆 🍸 CPU Page Faults																
[0] Graphics Device																
Unified Memory																
🗆 🍸 Data Migration (DtoH)			Data							Data						
- V GPU Page Faults		GPU Page Faults	GPU Pa	age Faults	GPU Pag G	iPU Page Faults	GPU Page	Faults	GPU Page F GPU	Pag GPU I	Page Faults	GPU Page	Faults	GPU F	age Faults	GPU Page
Gronagerautts												GPU Pa	ge Faults			
🗆 🍸 Data Migration (HtoD)														Data	Data	Data
Context 1 (CUDA)																
Compute							vectorA	dd(float consi	t *, float const *, flo	oat*, int)						
L 🍸 100.0% vectorAdd							vectorA	dd(float consi	t *, float const *, flo	oat*, int)						
Streams																
Default							vectorA	dd(float consi	t *, float const *, flo	oat*, int)						

FILTER AND ANALYZE Filtered

	233.5 ms	234 ms	234.5 ms	235 ms	235.5 ms	236 ms	236.5 ms	237 ms	237.5 ms	238 ms	238.5 ms	239 ms	239.5 ms	240 ms	240.5 ms	241 ms
Process "vecAdd_managed" (
Thread 3890149184																
Runtime API																
L Driver API																
Profiling Overhead																
Unified Memory																
- 🍸 CPU Page Faults																
[0] Graphics Device																
Unified Memory						_										
- 🍸 Data Migration (DtoH)			Data							Data						
🗅 🍸 GPU Page Faults										1						
Tota Migration (HtoD)														Dat	a Data	Data
Context 1 (CUDA)																
Compute																
L 🍸 100.0% vectorAdd										_						
Streams																
Default										_						
CPU Page Fault	s]		\mathbf{X}						
Access Type:	Re	ad	🗌 Write													
GPU Page Fault	S								Filte	ered i	nterva	ls				
Access Type:	Re	ead	🗌 Write		Atomic	□ P	refetch					1.5				
🛃 HtoD Migration	าร															
Reason:	🗌 Us	ser	Coher	ence	🗹 Prefetch	1										
🗹 DtoH Migration	าร															
Reason:	Us	ser	Coher	ence	🗹 Prefetch		viction									
Filter and Analyze																17 📀 I

UNOPTIMIZED APPLICATION

	29 s	.23 s 0.2	31 s 0.23	32 s 0.233 s	0.234 s	0.235 s	0.236 s	0.237 s	0.238 s	0.239 s	0.24 s	0.241 s 0.
Process "vecAdd_managed" (6162))											
Thread 3214706496						2.2 mc						
Runtime API												
Driver API												
Profiling Overhead												
 Unified Memory 												
- 🍸 CPU Page Faults												
🖃 🍸 Thrashing-Throttling										lemory	1 Thrac	hing
- 🍸 Memory Thrashing										ic mor y	inas	i i i s
[0] Graphics Device												
Unified Memory												
– 🍸 Data Migration (DtoH)												
- 🍸 GPU Page Faults	GPU Pa	ge Fa	GPU Page F	GPU Pag GPU	. GPU Pag GPU P	ag GPU Pa GPU P	. GP	PU Page	GPU P	age Fa		
- 🔽 Data Migration (HtoD)		GFOFage				GPU						
Thrashing-Throttling												
- T Memory Thrashing												
Context 1 (CUDA)							II.					
 Compute 												
Compute \$\screwty \formatty 100.0% vectorAdd(float)												
 Compute T 100.0% vectorAdd(float Streams 												
Compute Ty 100.0% vectorAdd(float Streams Default	-											
 Gompute □ 100.0% vectorAdd(float □ Streams □ Default 	•••											
 Compute ↓ ▼ 100.0% vectorAdd(float Streams L Default 												_
 Gompute ↓ ▼ 100.0% vectorAdd(float Streams Default ✓ Memory Thrashing 	***										C 11	_
 Compute □ 100.0% vectorAdd(float □ Streams □ Default ✓ Memory Thrashing ✓ CPU Page Faults 	***							Read	acces	s nage	faults	
 Compute ↓ ▼ 100.0% vectorAdd(float Streams Default ✓ Memory Thrashing ✓ CPU Page Faults Access Type: ▼ Reac 								Read	acces	s page	faults	
 Compute ↓ ♥ 100.0% vectorAdd(float Streams Default ✓ CPU Page Faults Access Type: ✓ React ✓ GPU Page Faults 								Read	acces	s page	faults	
 Compute [¬] 100.0% vectorAdd(float Streams [¬] Default Memory Thrashing [¬] CPU Page Faults Access Type:	d Write	Atomic	Prefetch					Read	acces	s page	faults	
 Compute ↓ ▼ 100.0% vectorAdd(float Streams Lefault ✓ Memory Thrashing ✓ CPU Page Faults Access Type: ✓ React ✓ GPU Page Faults Access Type: ✓ React ✓ HtoD Migrations 	d Write	Atomic	Prefetch					Read	acces	s page	faults	
 Compute T 100.0% vectorAdd(float Streams Default Memory Thrashing CPU Page Faults Access Type: ✓ Read GPU Page Faults Access Type: ✓ Read HtoD Migrations Reason: User 	d Write	Atomic	Prefetch					Read	acces	s page	faults	
 Compute T 100.0% vectorAdd(float Streams Default Memory Thrashing CPU Page Faults Access Type: S React GPU Page Faults Access Type: S React HtoD Migrations Reason: User DtoH Migrations 		Atomic ence Prefetc	Prefetch	,				Read	acces	s page	faults	
 Compute T 100.0% vectorAdd(float Streams Default CPU Page Faults Access Type: ✓ React GPU Page Faults Access Type: ✓ React HtoD Migrations Reason: User DtoH Migrations 	d Write	Atomic ence Prefetc	Prefetch					Read	acces	s page ad acce	faults	e faults
 Compute ∑ 100.0% vectorAdd(float Streams Default Ø CPU Page Faults Access Type: ♥ React Ø CPU Page Faults Access Type: ♥ React ♥ HtoD Migrations Reason: User DtoH Migrations Reason: User 	d Write	Atomic Atomic Prefetc ence Prefetc	Prefetch					Read Analy	acces	s page ad acce	faults ess pag	e faults
 Compute Compute Streams Default Memory Thrashing CPU Page Faults Access Type: S Read GPU Page Faults Access Type: Read HtoD Migrations Reason: User DtoH Migrations Reason: User 		Atomic ence Prefetc ence Prefetc	Prefetch					Read Analy	access /ze rea	s page ad acce	faults ess pag	e faults
 Compute Compute T 100.0% vectorAdd(float Streams Default Memory Thrashing CPU Page Faults Access Type: S Read GPU Page Faults Access Type: Read HtoD Migrations Reason: User DtoH Migrations Reason: User Filter and Analyze 	d Write	Atomic ence Prefetc ence Prefetc	Prefetch					Read Analy	access vze rea an	s page ad acce ad thra	faults ess pag shing	e faults
 Compute Compute T 100.0% vectorAdd(float Streams Default Memory Thrashing CPU Page Faults Access Type: S Read GPU Page Faults Access Type: Read HtoD Migrations Reason: User DtoH Migrations Reason: User Filter and Analyze The following table shows the start 		Atomic Atomic ence Prefetc ence Prefetc	Prefetch	Its after filtering				Read Analy	access vze rea an	s page ad acce ad thra	faults ess pag shing	e faults
 Compute Compute T 100.0% vectorAdd(float Streams Default CPU Page Faults Access Type: I React GPU Page Faults Access Type: I React HtoD Migrations Reason: User DtoH Migrations Reason: User Filter and Analyze The following table shows the state of the shows the shows the shows the shows the state of the shows the shows the shows the shows the shows the s		Atomic Atomic Prefetc Orefetc d memory migrati hsize Total CPU P	Prefetch h h Eviction ons and page fault rage faults Total of	lts after filtering GPU Page faults To	tal thrashing size	Total different pages		Read Analy	access vze rea an	s page ad acce ad thra	faults ess pag shing	e faults

OPTIMIZATION

OLD

int threadsPerBlock = 256; int numBlocks = (length + threadsPerBlock - 1) / threadsPerBlock;

kernel<<< numBlocks , threadsPerBlock >>>(A, B, C, length);

NEW

int threadsPerBlock = 256; int numBlocks = (length + threadsPerBlock - 1) / threadsPerBlock;

cudaMemAdvise(A, size, cudaMemAdviseSetReadMostly, 0); cudaMemAdvise(B, size, cudaMemAdviseSetReadMostly, 0);

kernel<<< numBlocks , threadsPerBlock >>>(A, B, C, length);

OPTIMIZED APPLICATION

💺 unoptimised.nvvp	*NewSession3	X										- 6
	33.25 ms	233.5 ms	233.75 ms	234 ms	234.25 ms	234.5 ms	234.75 ms	235 ms	235.25 ms	235.5 ms	235.75 ms	236 ms
Process "vecAdd_man	aged" (20	mc					
Thread 166942905	6					2.7						
Runtime API									cudaDeviceS	ynchronize		
Driver API												
Profiling Overhead												
Unified Memory												
🗏 🍸 CPU Page Faul	ts											
[0] Graphics Device												
Unified Memory												
🗆 🍸 GPU Page Fau	lts	GPU P	age Faults		GPU Page Faults	GPU Pa	ge Faults		GPU Page Fa	ults	GP	U Page Faults
🗏 🍸 Data Migratio	n (HtoD)		Data M					Data Mig	ration (HtoD)	Data Migration (HtoD)	D	ata Migration (HtoD)
Context 1 (CUDA)												
Compute						vectorAdd(float c	onst *, float const *, f	loat*, int)				
L 🍸 100.0% vec	torAdd					vectorAdd(float c	onst *, float const *, f	loat*, int)				
 Streams 												
L Default						vectorAdd(float c	onst *, float const *, f	loat*, int)				

No DtoH Migrations and thrashing

Speedup 4x (2.9 vs 12.2)

NVLINK PROFILING

Unguided Analysis

NVLINK VISUALIZATION

Visual Profiler



DGX-1V NVLINK TOPOLOGY



ogical NVLink P	Properties		-					
Logical NVLink	Peak Bandwidth	Physical NVLinks	Peer Access	System Access	Peer Atomic	System Atomic	Utilization %	Idle time %
GPU0<>GPU1	50 GB/s	1	Yes	No	Yes	No	0	51
GPU0<>GPU2	50 GB/s	1	Yes	No	Yes	No	0	50
GPU0<>GPU3	100 GB/s	2	Yes	No	Yes	No	0	47
GPU0<>GPU4	100 GB/s	2	Yes	No	Yes	No	0	50
GPU1<>GPU2	100 GB/s	2	Yes	No	Yes	No	0	51
GPU1<>GPU3	50 GB/s	1	Yes	No	Yes	No	0	47
GPU1<>GPU5	100 GB/s	2	Yes	No	Yes	No	0	48
GPU2<>GPU3	100 GB/s	2	Yes	No	Yes	No	0	48
GPU2<>GPU6	50 GB/s	1	Yes	No	Yes	No	0	48
GPU3<>GPU7	50 GB/s	1	Yes	No	Yes	No	0	46
GPU4<>GPU5	50 GB/s	1	Yes	No	Yes	No	0	50
GPU4<>GPU6	50 GB/s	1	Yes	No	Yes	No	0	50
GPU4<>GPU7	100 GB/s	2	Yes	No	Yes	No	0	48
GPU5<>GPU6	100 GB/s	2	Yes	No	Yes	No	0	49
GPU5<>GPU7	50 GB/s	1	Yes	No	Yes	No	0	46
GPU6<>GPU7	100 GB/s	2	Yes	No	Yes	No	0	49
ogical NVI ink T	broughput							
	Ave Through	hout Max	Through	hout Mi	Through	a ut		
	Avg Throug							
	34.381 M	MB/S	7.7190	GB/S	2.0281	(B/S		
	34.3701		1.130		2.0881	(B/S		
	34.98	MB/S	10.472		1.433 kB/s			
	35.494		11.7810		1.911 kB/s			
	40.005		11.000		4.1401			
	40.791		6 5 5 6 4		4.1701	(B/S		
	42.8011		6 412 (2 644			
	42.930	AD/S	11 762 (4 216			
	44.2211	AD/c	11.6767		4.2101			
	33 301 4	AB/c	10.662.0		1 0/01	B/s		
	33.5011	AB/c	10.002 (1 462			
	42 180		12 209 /		2 004			
	42.1091	AB/s	12 3/6 /		2.0041	BIS		
	42.435	AB/c	9.675		3 422 1	BIS		
JF02->GP03	40.251	MD/S	9.075 (00/5	3.4221			

NVLINK EVENTS ON TIMELINE

💺 *p2pNVLinkP2PBandwidthLa	atency_PASCAL_DGX.pdm 🛙					🗆 Properties 🛱 🗖
	11 s	11.5 s	12 s	12.5 s		NVlink
Process "p2pBandwidthLate					A	Start 4.71064 s (4,710,644,316 ns)
Thread 4028659584					Ξ.	End 72.71376 s (72,713,757,702 ns
Runtime API						Duration 68.00311 s (68,003,113,386 ns
Driver API		-				
Profiling Overhead	NVLink	Events on				
[0] Graphics Device		1.				
🖃 Nvlink	l 1r	neline				
[1] Graphics Device		۸				
[2] Graphics Device	^					
[3] Graphics Device						The nvlink interval color legend
 Context 8 (CUDA) 						Transmit Throughput
- 🍸 MemCpy (HtoD)						Receive Throughput
- 🍸 MemCpy (DtoH)						↑
– 🍸 MemCpy (DtoD)						
- 🍸 MemCpy (PtoP)						
	(4)					L
	I				_	
	MemCp	by API				Color Coding of NVLink Events

NVLINK ANALYSIS

Stage I: Data Movement Over PCIe

	0 s	0.05 s	0.1 s	0.15 s	0.2	s 0.25 s	216.10659 ms	0.35 s	0.4 s	0.45 s
Process "vectorAdd_naive" (12621)						_				
Thread 2896451328							216 millise	conds		
Runtime API			cudaMalloc			cudaMemcpy	cudaMemcpy	cudaMem	тсру	
L Driver API										
Profiling Overhead										
🖃 [0] GeForce GTX 1080										
Context 1 (CUDA)										
- 🍸 MemCpy (HtoD)						Memcpy HtoD [sync]	Memcpy HtoD [sync]			
- 🍸 MemCpy (DtoH)								Memcpy Dto	oH [sync]	
🗏 Compute										
- 🍸 100.0% vectorAdd(float										
Streams										
Default						Memcpy HtoD [sync]	Memcpy HtoD [sync]	Memcpy Dto	oH [sync]	

NVLINK ANALYSIS

Stage II: Data Movement Over NVLink



NVLINK ANALYSIS

Stage III: Data Movement Over NVLink with Streams



INSTRUCTION LEVEL PROFILING (PC SAMPLING)

PC SAMPLING

PC sampling feature is available for device with CC >= 5.2

Provides CPU PC sampling parity + additional information for warp states/stalls reasons for GPU kernels

Effective in optimizing large kernels, pinpoints performance bottlenecks at specific lines in source code or assembly instructions

Samples warp states periodically in round robin order over all active warps

No overheads in kernel runtime, CPU overheads to parse the records

PC SAMPLING UI

Pie chart for sample distribution for a CUDA function

Sample distribution





Source-Assembly view

30 📀 nvidia

MPI PROFILING

MPI PROFILING

\$ mpirun -n 4 nvprof --process-name "MPI Rank %q{OMPI COMM WORLD RANK}" -context-name "MPI Rank %q{OMPI_COMM_WORLD_RANK}" -o timeline.%q{OMPI_COMM_WORLD_RANK}.pdm ./simpleMPI Running on 4 nodes ==21977== NVPROF is profiling process 21977, command: ./simpleMPI ==21983== NVPROF is profiling process 21983, command: ./simpleMPI ==21979== NVPROF is profiling process 21979, command: ./simpleMPI ==21982== NVPROF is profiling process 21982, command: ./simpleMPI <program output> ==21982== Generated result file: timeline.0.pdm ==21977== Generated result file: timeline.3.pdm ==21983== Generated result file: timeline.1.pdm ==21979== Generated result file: timeline.2.pdm

MPI PROFILING nvprof daemon mode

Shell 1

\$ nvprof --profile-all-processes
-o out.%h.%p.%q{OMPI_COMM_WORLD_RANK}
<nvprof listens in daemon mode>

<profiling data is generated>

Shell 2



\$ mpirun -n 4 ./simpleMPI

MPI PROFILING Importing into the Visual Profiler

File View Window Help	J	
New Session Ctrl+N Open Ctrl+O Save All Shift+Ctrl+S Import Ctrl+i		Import Nyprof Data Import Profile Data for Multiple Processes Select nyprof profile files containing timeline data for multiple processes
2 Exit	3	Profile Files Timeline Options Connection:
Import Select Import profile data generated by nvprof.	Import Nvprof Data Nvprof profile files Import profile data for a single process or for multiple processes	/home/apoorvaj/sw/gpgpu/bin/x86_64_Linux_debug/timeline.3.pdm /home/apoorvaj/sw/gpgpu/bin/x86_64_Linux_debug/timeline.2.pdm /home/apoorvaj/sw/gpgpu/bin/x86_64_Linux_debug/timeline.1.pdm /home/apoorvaj/sw/gpgpu/bin/x86_64_Linux_debug/timeline.0.pdm
Select an import source: type filter text Image: Command-line Profiler	 Single process Multiple processes 	
Nvprof		Normalize each profile file independently
		Use fixed width segments for Unified memory timeline
		Number of segments Specify the number of segments for unified memory timelines [default 100]
< Back Next > Cancel Finish	< Back Next > Cancel Finish	< Back Next > Cancel Finish

MPI PROFILING Visual Profiler



MPI + NVTX

Manual mode

```
nvtxEventAttributes_t range = {0};
range.message.ascii = "MPI_Scatter";
nvtxRangePushEx(range);
int result = MPI_Scatter(...);
nvtxRangePop();
```

Interception mode



Auto-generate mpi_interception.so

https://devblogs.nvidia.com/parallelforall/gpupro-tip-track-mpi-calls-nvidia-visual-profiler/



LD_PRELOAD=mpi_interception.so



Run your MPI app with nvprof. MPI calls will be auto-annotated using NVTX.

MPI PROFILING Interception



REMOTE PROFILING

NVVP: MULTI-HOP REMOTE PROFILING



NVVP: MULTI-HOP REMOTE PROFILING One-Time Setup



Configure script on the login node

Script	
Login Node	\mathcal{I}



😣 🗉 🛛 New R	emote Connection	
Remote Con	nections	
Manage avail	able connections	
tk@10.24.204	1.242	Add
		Remove
Host name:	10.24.204.242	
User name:	tk	
Label:	tk@10.24.204.242	
System type:	SSH 22	
	Cancel	inish



 Toolkit path: 	/home/tk/data/p4ws, Browse	Detect
Library paths:	🖶 Add new path	Browse
		Delete
		(Q1
Custom Script:	/home/tk/remote_profiling.pl	Browse
Temporary Location:	/tmp	Browse

NVVP: MULTI-HOP REMOTE PROFILING Application Profiling



Connection:	tk@10.24.204.242	nnections
Toolkit/Script:	/home/tk/remote_profiling.pl	Manage
File:	/home/compute_node/apps/matrixmul	Browse
Working directory:	Enter working directory [optional]	Browse
Arguments:	Enter command-line arguments	
	Profile child processes	÷
Environment:	Name Value	Add
		Delete



Application transparently runs on compute node and profiling data is displayed in the Visual Profiler

8	🔍 🗉 NVIDIA Visual Profiler						
*		Q Q I E F K					
8	💺 *NewSession1 🛙						5
		0.36 s	0.365 s	0.37 s	0.375 s	O .	
	Process "matrixmul" (27514)						
===	Thread 1003218752						
	Runtime API		C	udaMalloc cudaMal	loc cudaFree	cudaFre	
	Driver API						
	Profiling Overhead						
	[0] GeForce GTX TITAN X						
	Context 1 (CUDA)						
	- 🍸 MemCpy (HtoD)						
	- 🍸 MemCpy (DtoH)						
	Compute						
	– 🍸 100.0% dmatrixmu						
	Streams						
	Default						
		(4)					

VOLTA SUPPORT

VOLTA SUPPORT

	0.775 s	0.7775 s	0.78 s	0.7825 s	0.785 s	0.7875 s	0.79 s	0.7925 s	0.795 s	0.7975 s	0.8 s	0.8025 s	0.805 s
Process "blackscholes2" (181													
Thread 379262784													
Runtime API	cudaMemcpy cudaMemcpy cudaMemcpy cudaMemcpy cuda									cudaMemcpy			
L Driver API													
Profiling Overhead													
[0] Graphics Device													
Context 1 (CUDA)													
- 🍸 MemCpy HtoD)	Memcpy Hto	. Memcpy Hto N	Aemcpy Hto										
🗏 🍸 MemCpy DtoH)											Memcpy Dto M	1emcpy Dto	
Compute													
⊢ 🍸 98.0% GPUBlackSc													
└────────────────────────────────────													
- 🍸 1.0% C PUBlackSch													
Streams													
L Default	Memcpy Hto	. Memcpy Hto N	Aemcpy Hto								Memcpy Dto M	1emcpy Dto	
🕒 Stream 14													
Stream 1:													

GPU Trace

GV100 Device Attributes

🔲 Properties 🛱		
[0] Graphics Device		
GPU UUID	GPU-2fa3ace0-e500-84d5-d455-7d82b2714628	P
▼ Duration		
Seccion	1 51796 c (1 517 957 989 pc)	
▼ Attributes		
Compute Capability	7.0	
▼ Maximums		
Threads per Block	1024	
Threads per Multiprocessor	2048	
Shared Memory per Block	48 KiB	
Shared Memory per Multiprocessor	96 KiB	
Registers per Block	65536	
Registers per Multiprocessor	65536	
Grid Dimensions	[2147483647, 65535, 65535]	
Block Dimensions	[1024, 1024, 64]	
Warps per Multiprocessor	64	

OTHER IMPROVEMENTS

OTHER IMPROVEMENTS

Tracing and profiling of Cooperative Kernel launches is supported

The Visual Profiler supports remote profiling to systems supporting ssh algorithms with a key length of 2048 bits

OpenACC profiling is now supported on systems without CUDA setup

nvprof flushes all profiling data when a SIGINT or SIGKILL signal is encountered

REFERENCES

NVIDIA toolkit documentation: <u>http://docs.nvidia.com</u>

CUDA Profiler User Guide: <u>http://docs.nvidia.com/cuda/profiler-users-guide/index.html</u>

Other GTC 2017 sessions:

- S7824 DEVELOPER TOOLS UPDATE IN CUDA 9
- S7519 DEVELOPER TOOLS FOR AUTOMOTIVE, DRONES AND INTELLIGENT CAMERAS APPLICATIONS
- S7445 WHAT THE PROFILER IS TELLING YOU: OPTIMIZING WHOLE APPLICATION PERFORMANCE

S7444 - WHAT THE PROFILER IS TELLING YOU: OPTIMIZING GPU KERNELS

