

NVIDIA PROFILING TOOLS

Jeff Larkin, December 04, 2018

UPDATES FOR CUDA 9.2

NVPROF	VISUAL PROFILER
Many New Metrics: - Tensor Core Metrics - L2 Metrics - Memory Instructions Per Load/Store	Summary View for Memory Hierarchy Improved Handling of Segments for UVM Data on the Timeline
Display PCIe Topology	
View Trace and Profile in combined output (trace)	

UPDATES FOR CUDA 10.0

Added tracing support for Turing

New kernel profiler - Nsight Compute (supports Turing)

OpenMP profiling

Tracing support for CUDA kernels, memcpy and memset nodes launched by a CUDA Graph

Support for version 3 NVIDIA Tools Extension API (NVTX) (This is a headeronly implementation)



CUDA VISUAL PROFILER

Overview of key features

Kernel profile - memory hierarchy view

Unified Memory

NVLink

PC sampling

OpenACC/OpenMP Profiling

NVTX



NVIDIA'S VISUAL PROFILER (NVVP)

Timeline 🖃 [0] Tesla K40c Context MPS (CUDA) MemCpy (HtoD) - 🕎 MemCpy (DtoH) float const ... Step10 cuda .. Step10 cuda kernel. Step10 d Compute Step10 cuda k... Step10 cuda.. Step10 cuda kernel(int.. float const ... Step10 cuda .. Step10 cuda kernel.. Step10 d └ 🍸 100.0% Step10 c... Step10 cuda.. Step10 cuda kernel(int.. Step10 cuda k.. Streams



System

1. CUDA Application Analysis

2. Performance-Critical Kernels

3. Compute, Bandwidth, or Latency Bound

Guided

The first step in analyzing an individual kernel is to determine if the performance of the kernel is bounded by computation, memory bandwidth, or instruction/memory latency. The results at right indicate that the performance of kernel "Step10_cuda_kernel" is most likely limited by compute.

🕕 Perform Compute Analysis

The most likely bottleneck to performance for this kernel is compute so you should first perform compute analysis to determine how it is limiting performance.

4 Perform Latency Analysis

👞 Perform Memory Bandwidth Analysis

Instruction and memory latency and memory bandwidth are likely not the primary performance bottlenecks for this kernel, but you may still want to perform those analyses.

4 Rerun Analysis

If you modify the kernel you need to rerun your application to update this analysis.

DATA MOVEMENT IN VISUAL PROFILER

🙍 Po	GPROF @jlarkin-dt	and a second						x
<u>F</u> ile	<u>∨</u> iew <u>W</u> indow <u>R</u> un <u>H</u> elp							
*	🖕 🖳 📑 👒 • [🕂 🔾	🔍 F 🥆 🔣 🚑 🚇	[<u>Å</u> •					
8	*NewSession1 🛿						- 0	8
		35 s	35.5 s	36 s	36.5 s	37 s		: 🗆
	🖃 Process "cg" (23919)			I				-
	🖃 Thread 3484149632							
	└ OpenACC	acc_co acc	_enter_data@vec acc_ acc_	co acc_enter_	_data@vec acc_co acc_wa	acc_enter_d	ata@vec.	
	L Driver API	cuStre	cuSi	tre	cuStre			
	Profiling Overhead							
	📃 [0] Tesla K20c							
	🖃 Context 1 (CUDA)							
	└ 🍸 MemCpy (HtoD)							
	└ 🍸 MemCpy (DtoH)							
	🖃 Compute	_Z6ma	_Z6	ma	_Z6ma			
	└ 🍸 97.0% _Z6matvec	_Z6ma	_Z6	ma	_Z6ma			
	└ 🍸 2.0% _Z6waxpbyd							
	└ 🍸 0.9% _Z3dotRK6v							
	└ 🍸 0.1% _Z3dotRK6v							
	🖃 Streams							
	L Stream 13	_Z6ma	_Z6	na	Z6ma			
		•						Þ

UVM IN VISUAL PROFILER

K NVIDIA Visual Profiler				-			-		x
<u>File View W</u> indow <u>Run H</u> elp									
	Щ =, %, + ⊕ ∈	2 🔍 F 🥆 📙	J 📮 📇 🗛 🔻						
🖻 🔍 *timelineUM.nvprof 🛛									B
	0.7 s 0.8 s	s 0.9 s	1 s	1.1 s	1.2 s	1.3 s	1.4 s	1.5 s	
Process "cg" (64759)									
🖃 🖃 Thread 299904									
- OpenACC			acc_compute_co	nstruct@vector.h:12		acc	acc acc	acc acc ac	C (
L Driver API			cuStream	Synchronize		cuSt	cuSt cuSt	cuSt cuSt cu	ut
L Profiling Overhead									
 Unified Memory 									
- 🍸 CPU Page Faults	a CPU Pa CPU Pa	a							
[0] Tesla P100-SXM2-16GB									
Unified Memory									
🗆 🍸 GPU Page Faults	GPU P	a <mark>GPU Pa GPU Pa</mark> .	GPU Pa GPU Pa.	GPU Pa GPU Pa	GPU Pa (GPU Pa GPU Pa			
🗆 🍸 Data Migration (HtoD)	Data N	/I Data M Data M.	Data M Data M.	Data M Data M	Data M I	Data M Data M			
 Context 1 (CUDA) 									
🗏 🍸 MemCpy (HtoD)									
🗆 🍸 MemCpy (DtoH)									
Compute			_Z6matvecRK6matr	xRK6vectorS4_12_g	ри				
└ 🍸 95.6% _Z6matvecRK			_Z6matvecRK6matr	xRK6vectorS4_12_g	ри				
└ 🍸 3.3% _Z6waxpbydR									
└ 🍸 0.7% _Z3dotRK6vect									
└ 🝸 0.4% _Z3dotRK6vect									
 Streams 									
└─ Stream 13			_Z6matvecRK6matr	xRK6vectorS4_12_g	pu				
	•	III						ł	

KERNEL PROFILE

Memory hierarchy view



Segment mode timeline

¶ *NewSession1 ⊠																■ Properties 🛱		
Process "Jacobi" (5267) Thread 655341440 Runtime API Driver API Profiling Overhead Unified Memory TY CPU Page Faults [0] Graphics Device Unified Memory TY Data Migration (CY GPU Page Faults CY Data Migration CY DAta Migratio CY DAta Migratio CY DAta Migra	DtoH)	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, 5	0.75 s	0.8 s	0.85 s	CPU Page Faults The segment mode is used for into equal width segments a segment are shown. Timestamp End Duration Virtual Address Range Process	for this timeline. In this mode the tin and only aggregated data values for 268.60478 ms (268,6 278.93573 ms (278,9 10.33095 ms (10,330 0x900000000 - 0x90 5267	neline is split each time 04,778 ns) 35,731 ns) ,953 ns) 01ff000
Context 1 (CUDA) Compute U 10.0% jacobi Streams Default	Seg	ţmei	nt m	ode	inte	erval		H	eat pa	map ige f	for (aults	CPU				The number of CPU page fa 0-10% [0-30000 10-20% [30000-6 20-30% [60000-9 30-40% [90000-1 40-50% [120000- 50-60% [150000- 60-70% [180000- 70-80% [210000- 80-90% [240000- 90-100% [>270000	oults per second within the segment [50000] [50000] [20000] [120000] [150000] [180000] [210000] [240000] [240000] [270000] [0]	

Switch to non-segment view

Uncheck	Select settings view	
🗔 Analysis 🔚 GPU Details (Summary) 🖽 CPU Details 🗖 OpenACC I	Details 💷 Console 🗔 Settings 🕴	
Session NewSession1		
Executable Use fixed width segments for Unified memo	ory timeline	
Timeline Options Number of segments 100		
Verifatysis (
Start time 225	ms to End time 700	ms
Enable timelines in session view		
V 🗹 All		
▼ Ø Process		
Select this tab	Load data within a specific time range	
S OpenAcc		
CPU Page Faults		
V S Device		
 Image: Second se		
✓ 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 🛛 💺 *Ne	wSession1-clone ¤									- 0
	0.25 s	0.3 s	0.35 s	Selected	interval	0.5 s	Source	location	0.65 s	0.7 s
Process "jacobi" (5267)								(ocacion		
Thread 655341440										
Runtime API										
L Driver API										
Profiling Overhead										
 Unified Memory 										
- 🍸 CPU Page Faults										
[0] Graphics Device					<u> </u>					
Unified Memory										
🗆 🍸 Data Migration (Dto	H)									
🗆 🍸 GPU Page Faults										
- 🍸 Data Migration (Hto	C)									
 Context 1 (CUDA) 										
Compute										
L 🍸 100.0% jacobi_ite	r									
Streams										
L Default										

Properties X	
CPU Page Faults	
Timestamp	440.45958 ms (4 <mark>40,459,581 ns)</mark>
Memory Acccess Type	Write
Virtual Address	0x900100000
Source Location	main@jacobi.cu:130
Process	25684

12 📀 nvidia

CPU Page Fault Source Correlation

■ Properties	
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
Memory Access Type Virtual Address Source Location Process	Write 0x900100000 main@jacobi.cu:130 25684

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[(ny-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
```

VISUAL PROFILER - NEW UNIFIED MEMORY EVENTS

Page throttling, Memory thrashing, Remote map



Filter and Analyze

	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						
- V CPU Page Faults										· · · · •						
Grorageradics																
🗆 🍸 Data Migration (HtoD)														Data	Data	Data
Context 1 (CUDA)								- - - - - - - - -								
 Compute 										_						
L 🍸 100.0% vectorAdd																
 Streams 																
Default										_						
										\rightarrow						
				1				1		\mathbf{N}						
CPU Page Faul	ts															
		ad	🗌 Write													
Ассезь турс.				·						N						
GPU Page Faul	ts								E214							
		ad		r	Atomic		ofotob		F110	erea i	nterva	lS				
Access Type.		au	write		Atomic		eretti									
🖌 HtoD Migratio	ns															
Reason:	Us	er	Coher	rence	Prefetch	1										
V DtoH Migratio	ns															
Reason:	Us	er	Coher	rence	🗹 Prefetch		lction									
Filter and Analyze																15 💿 ୮

NVLINK visualization

Unguided Analysis

🖥 An lysis 🛱 🔤 GPU Details (Summary) 🖽 CPU D	tails 🗖 OpenACC Details 📮 Console 🗔 Settings	Static	Runtime
E Reset All Analyze All Consider the stages select a host- aunched kernel instance in the timeline. pplication	i NVLink Analysis The following NVLink topology diagram shows logical NVLink connections between GPUs and CPUs. A logical NVLink can contai receive throughput of device A is same as the transmit throughput of device B. The tables on right hand side show the propertie	properties in one or more physical links. When two devices / es for each logical NVLink	Values
Data Movement And Concurrency 📀	* NVLink utilization may vary in accuracy, because any activity within the sampling period is treated as active, even though most	: of that period could be <mark>i</mark> dle. IVLink Properties	1
Compute Utilization	CPU1 CPU3	NVLink Peak Physical Peer System Bandwidth NVLinks Access Access	Peer System Atomic Atomic Utilization % Idle time %
Cernel Performance	Tesla P100-SX GPU0< GPU0<	->CPU0 80 GB/s 2 No Yes ->GPU1 80 GB/s 2 Yes No	No No 0 1 Yes No n/a 10
VLink	138.21 MB/s 0 B/s 0 B/s 142.06 MB/s GPU1< 90.78 MB/s 96.86 MB/s GPU2<	->CPU0 80 GB/s 2 No Yes ->CPU1 80 GB/s 2 No Yes	No No 0 1 No No 0 1
Inified Memory	CPU 0 CPU 1 GPU2<	->GPU3 80 GB/s 2 Yes No ->CPU1 80 GB/s 2 No Yes	Yes No n/a 100 No No 0 1
Option to collect	90.92 MB/s 0 B/s 0 B/s 0 B/s Logical N	IVLink Throughput	· · · · · ·
VLink information	ISO,00 MD/S GPU 0 Tesla P100-SX Tesla P100-SX ISO MD/S GPU 2 Tesla P100-SX ISO MD/S GPU 2 Tesla P100-SX	NVLink Avg Throughput Max Throughput Min CPU0 90.917 MB/s 36.085 GB/s 36.085 GB/s	Throughput 5.691 kB/s
	GPU0->	-GPU1 0 B/s 0 B/s -GPU1 0 B/s 0 B/s	0 B/s
Version	NVLink version 1.0 GPU1-> GPU1->	CPU0 90.777 MB/s 36.031 GB/s -CPU0 138.241 MB/s 33.14 GB/s	5.847 kB/s 1.949 kB/s
	Bandwidth usade by NVLink GPU2-> 90 - 100 % GPU2->	CPU1 96.692 MB/s 14.798 GB/s -CPU1 141.799 MB/s 16.495 GB/s	14.791 kB/s 5.764 kB/s
	70-80% Topology Selected GPU2->	GPU3 0 B/s 0 B/s -GPU3 0 B/s 0 B/s	0 B/s 0 B/s
	50-60% 40-50% 30.40%	CPU1 96.856 MB/s 14.786 GB/s -CPU1 142.063 MB/s 16.497 GB/s	13.863 kB/s 1.764 kB/s
olor codes for	20 - 30 % 10 - 20 %		
NVI ink	0-10%		

NVLink events on timeline



Multi-hop remote profiling - Application Profiling



Connection:	tk@10	.24.204.242 ‡ Manage con	nnections						
Toolkit/Script:	/home	home/tk/remote_profiling.pl							
File:	/home	/compute_node/apps/matrixmul	Browse						
Working directory:	Enter v	Enter working directory [optional] Brows Enter command-line arguments							
Arguments:	Enter c								
	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									
*	🖆 🔜 🖳 🖳 🗣 🗣 🛨 🗨 🔍 🖃 F 🥆 🔣 🚆 🚨 🦾 🔻									
Ð	♦ *NewSession1 ☎									
		0.36 s	0.365 s	0.37 s	0.375 s	0.				
	Process "matrixmul" (27514)									
m	Thread 1003218752									
	Runtime API		c	udaMalloc cudaMal	loc cudaFree	cudaFre				
	L Driver API									
	Profiling Overhead									
	[0] GeForce GTX TITAN X									
	Context 1 (CUDA)									
	- 🍸 MemCpy (HtoD)									
	– 🍸 MemCpy (DtoH)									
	 Compute 									
	- 🍸 100.0% dmatrixmu									
	 Streams 									
	Default									
		(4)								



CPU SAMPLING

- CPU profile is gathered by periodically sampling the state of each thread in the running application.
- The CPU details view summarizes the samples collected into a call-tree, listing the number of samples (or amount of time) that was recorded in each function.

VISUAL PROFILER CPU Sampling



20 📀 **DVIDIA**

Selected thread

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

VISUAL PROFILER - PC SAMPLING Option to select sampling period

🗔 Analysis 🛅 GPU D	Details (Summary) 🏦 CPU Details	🗖 OpenACC Details 📮 Console 🗔 Settings 🛿	 E
Session NewSess	ion1		
Executable Timeline Options	PCIe Override: Rerun analysis afte Device: [0] Graphics Device \$	r updating	P
Analysis	PCIe Generation: 2	Override: Enter PCIe generation to use for analysis, allowed values are 2, 3 [optional, clear to use default]	
	PCIe Link Width: 4	Override: Enter PCIe link width to use for analysis, allowed values are 1, 2, 4, 8, 16, 32 [optional, clear to use default]	
	PCIe Link Rate: 5 Gbit/s	Override: Enter PCIe link rate to use for analysis in Mbits/s [optional, clear to use default]	
	Sampling period: Rerun Kernel Pro	Image: Select sampling period Image: Select sampling period Image: Select sampling period will be in 2^n cycles	

PC SAMPLING UI

Pie chart for sample distribution for a CUDA function

Sample distribution





Source-Assembly view

23 📀 nvidia

MULTI-PROCESS PROFILING

When running nvprof with multiple processes, it's useful to label each process:

\$ nvprof -o timeline rank%q{OMPI COMM WORLD RANK} \

--context-name "MPI Rank %q{OMPI_COMM_WORLD_RANK} \

--process-name "MPI Rank %q{OMPI_COMM_WORLD_RANK} \

--annotate-mpi openmpi ...

MPI PROFILING Importing into the Visual Profiler

	······································	
File View Window Help New Session Ctrl+N Open Ctrl+O		4 © Import Nyprof Data
Save All Shift+Ctrl+S Import Exit		Import Profile Data for Multiple Processes Select nvprof profile files containing timeline data for multiple processes
2	3	Profile Files Timeline Options Connection: The nvprof profile files:
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 original	/home/apoorvaj/sw/gpgpu/bin/x86_64_Linux_debug/timeline.3.pdm Browse /home/apoorvaj/sw/gpgpu/bin/x86_64_Linux_debug/timeline.2.pdm Remove /home/apoorvaj/sw/gpgpu/bin/x86_64_Linux_debug/timeline.0.pdm Remove
Select an import source: type filter text Image: Command-line Profiler Nvprof Nvprof	Single process Multiple processes	
		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



26 📀 nvidia.

PROFILER API

Real applications frequently produce too much data to manage.

Profiling can be programmatically toggled:

```
#include <cuda_profiler_api.h>
cudaProfilerStart();
```

```
cudaProfilerStop();
```

...

This can be paired with nvprof:

```
$ nvprof --profile-from-start off ...
```

SELECTIVE PROFILING

When the profiler API still isn't enough, selectively profile kernels, particularly with performance counters.

\$ nvprof --kernels :::1 --analysis-metrics ...

context:stream:kernel:invocation

Record metrics for only the first invocation of each kernel.

NVTX ANNOTATIONS

The NVIDIA Tools Extensions (NVTX) allow you to annotate the profile:
 #include <nvToolsExt.h> // Link with -lnvToolsExt
 nvtxRangePushA("timestep");
 timestep();

```
nvtxRangePop();
```

See <u>https://docs.nvidia.com/cuda/profiler-users-guide/index.html#nvtx</u> for more features, including V3 usage.

NVTX IN VISUAL PROFILER



30 📀 nvidia.

EXPORTING DATA

It's often useful to post-process nvprof data using your favorite tool (Python, Excel, ...):

It's often necessary to massage this file before loading into your favorite tool.

OpenAcc->Driver API->Compute correlation

OPENACC PROFILING



OpenAcc timeline

OPENMP PROFILING

Information about OpenMP regions using the OpenMP tools interface (OMPT) starting CUDA 10.0

Supported on x86_64 and Power Linux with PGI runtime 18.1+

Supported added in the CUPTI, nvprof and Visual Profiler

OPENMP PROFILING IN NVPROF

nvprof option openmp-profiling to enable/disable the OpenMP profiling, default on

nvprof openmp-profiling on ./omp-app										
	Туре	Time(%)	Time	Calls	Avg	Min	Max M	Name		
OpenMP	(incl):	99.97%	277.10ms	20	13.855ms	13.131ms	18.151ms	omp_parallel		
		0.03%	72.728us	19	3.8270us	2.9840us	9.5610us	omp_idle		
		0.00%	7.9170us	7	1.1310us	1.0360us	1.5330us	omp_wait_barrier		
Optionprint-openmp-summary to print a summary of all recorded OpenMP										

\$

activities

OPENMP PROFILING IN VISUAL PROFILER

🕵 NVIDIA Visual Profiler												- • •
File View Window Run Help												
💺 *openmp.prof 🔀												- 8
	0 s 0.01	s 0.02 s	0.03 s 0	.04 s 0.0	5 s 0.06 s	0.07 s	0.08 s	0.09 s	0.1 s	0.11 s	0.12 s	0.13 s
Process 0												
Thread 0												
- OpenMR	OMP_P OMP	Par OMP_Par OM	P_Pa OMP_P (OMP_P OMP_P	OMP OMP_	OMP OMP						
- Openimp												
🔄 Analysis 🔛 GPU Details (Summ	ary) 🏢 CPU De	tails 📺 OpenACC Det	ails 🛛 🚛 OpenMP L	Details 💥 🚊 Co	insole 📑 Setting	5				🗏 Properties 💥		
										OMP_Parallel		
Name	%	Time	Calls							Start		51.90515 ms
OMP Parallel	99.976%	123.33896 ms	20							End		58.14554 ms
OMP_Wait_barrier	0.331%	0.40816 ms	19							Duration		6.2404 ms (6
OMP_Idle	0.074%	0.0907 ms	19									

OPENMP PROFILING IN VISUAL PROFILER Table View

🗔 Analysis 🔜 GPU Details (Summary)	CPU Details	🏢 OpenACC Detail	s 🕞 OpenMP Deta	ails 🛛	📃 Console	Settings		
Name	%	Time	Calls					
OMP_Parallel	93.895%	5.4 s	3003					
OMP_Idle	16.619%	0.9 s	3002					
OMP_Wait_barrier	7.528%	0.4 s	3001					

PROFILING NVLINK USAGE

Using nvprof+NVVP

Run nvprof multiple times to collect metrics

jsrun <args> nvprof --output-profile profile.<metric>.%q{OMPI_COMM_WORLD_RANK} \

--aggregate-mode off --event-collection-mode continuous

--metrics <metric> -f

Use `--query-metrics` and `--query-events` for full list of metrics (-m) or events (-e)

Combine with an MPI annotated timeline file for full picture

SUMMIT NVLINK TOPOLOGY

Results

i NVLink Analysis

The following NVLink topology diagram shows logical NVLink connections between GPUs and CPUs. A logical NVLink can contain one or more physical links. When two devices A and B are connected by an NVLink, the receive throughput of device A is same as the transmit throughput of device B. The tables on right hand side show the properties for each logical NVLink.

* NVLink utilization may vary in accuracy, because any activity within the sampling period is treated as active, even though most of that period could be idle.



Logical NVLink	PeakBandwidth	PhysicalNVLinks	PeerAccess	SystemAccess	PeerAtomic	SystemAtomic	Utilization %	lo
GPU0<>CP	100 GB/s	2	No	Yes	No	Yes	0	
GPU0<>GP	100 GB/s	2	Yes	No	Yes	No	0	
GPU0<>GP	100 GB/s	2	Yes	No	Yes	No	0	
GPU0<>GP	84 GB/s	2	Yes	No	Yes	No	0	
GPU0<>GP	84 GB/s	2	Yes	No	Yes	No	0	
GPU0<>GP	84 GB/s	2	Yes	No	Yes	No	0	
GPU1<>CP	100 GB/s	2	No	Yes	No	Yes	0	
GPU1<>GP	100 GB/s	2	Yes	No	Yes	No	0	
GPU1<>GP	84 GB/s	2	Yes	No	Yes	No	0	
GPU1<>GP	84 GB/s	2	Yes	No	Yes	No	0	
GPU1<>GP	84 GB/s	2	Yes	No	Yes	No	0	
GPU2<>CP	100 GB/s	2	No	Yes	No	Yes	0	
GPU2<>GP	84 GB/s	2	Yes	No	Yes	No	0	
GPU2<>GP	84 GB/s	2	Yes	No	Yes	No	0	
GPU2<>GP	84 GB/s	2	Yes	No	Yes	No	0	
GPU3<>CP	100 GB/s	2	No	Yes	No	Yes	0	
GPU3<>GP	100 GB/s	2	Yes	No	Yes	No	0	
GPU3<>GP	100 GB/s	2	Yes	No	Yes	No	0	
GPU4<>CP	100 GB/s	2	No	Yes	No	Yes	0	
GPU4<>GP	100 GB/s	2	Yes	No	Yes	No	0	
GPU5<>CP	100 GB/s	2	No	Yes	No	Yes	0	
<								>

Logical NVLink Properties

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

