

HPC SOFTWARE – DEBUGGER AND PERFORMANCE ANALYSIS TOOLS

NOVEMBER 13, 2024 I MICHAEL KNOBLOCH I M.KNOBLOCH@FZ-JUELICH.DE



Mitglied der Helmholtz-Gemeinschaft





WHY SHOULD YOU CARE ABOUT TOOLS?





NEW APPLICATION?





WORKING WITH LEGACY CODES?





VETERAN HPC USER, BUT NEW TO JSC?



• Assess performance on a JSC machine



 Compare behavior on different machines



Investigate scaling behavior



			Name	Туре	Value
Thread State	TID		_ ≙	int	0x00000000000
Breakpoint	1.1	1.1 tensorflow::SoftmaxXentWith nstar int		0x0000006 (6)	
Stopped	1.2	pthread_cond_wait	grap	int	0x00000015 (21)
Stopped	1.3	pthread cond wait	[Add		
Stopped	1.4	othread cond wait			
Copped		Princing_cons_mar			
Select process or thread attributes to group by:					
/ Thread State					
Function					
Process State					
Control Group	5				
+		2			
tion Points ×	Comma	nd Line 🗶			
ID❤	Туре	Stop Location		L	ine

DEBUGGER & CORRECTNESS TOOLS



WHAT IS DEBUGGING?





DEBUGGING TOOLS (STATUS: NOV 2024)

- Debugger:
 - CUDA-GDB
 - TotalView
 - LinaroForge DDT
- Memory Analyzer:
 - Intel Inspector
 - Archer
- Correctness Checker:
 - MUST



I DON'T KNOW WHERE YOU ARE, I DON'T KNOW HOW YOU WORK, BUT I WILL FIND YOU AND I WILL FIX YOU



CUDA-GDB



- Part of the CUDA toolkit
- Extension to gdb
- CLI and GUI (Nsight)
- Simultaneously debug on the CPU and multiple GPUs
- Use conditional breakpoints or break automatically on every kernel launch
- Examine variables, read/write memory and registers
- Inspect GPU state when the application is suspended
- Identify memory access violations

] 🗈 🕶 📾] 🕸 = O = 9 = 9 = 9 = 1 × × × × × × × × × ×								
🕸 Debug 🛛 💦 🙀 🙀 😨 🖓 🖬 🐨 🖓 🖬 🖗 Variables 💁 Breakpoints 💽 CUDA 🕄 📑 Modules 🖓 🖓								
🔻 🔊 vectorAdd {0} [device: gk110 (0)] (Breakpoint)								
CUDA Thread (0,0,0) Block (0,0,0)								
CUDA Thread (1,0,0) Block (0,0,0)		Search CUDA Information						
🔻 🚳 All CUDA Threads	▼ 🎲 (0),0,0)	SM 11	256 threads of 256 are runr				
🔻 🍓 Block (0,0,0) [sm: 11]	ø	(0,0,0)	Warp 0 Lane 0	Warp 0 Lane 0 🖻 vectorAdd.cu:36 (0x9a6!				
CUDA Thread (0,0,0) [warp: 0 lane: 0] (vectorAdd.cu:36)	្ 💣	(1,0,0)	Warp 0 Lane 1	vectorAdd.cu:36 (0x9a6530				
e vectorAdd.cu ສ	- 0	E Outline	배 Registers 없	* *** •** E 📬 🛃 🎽 💆				
33 {		Name	T(0,0,0)B(0,0,0) T(1,0,0)B(0,0,0)				
<pre>34 int i = blockDim.x * blockIdx.x + threadIdx.x;</pre>		1818 R5	4	4				
35 26 if (i < numElements)		1919 R6	3149824	3149824				
37 {		1919 R7	4	4				
38 $C[i] = A[i] + B[i];$		1111 R8	0	1				
39 } 40 }		1111 R9	0	1				
41	¥	배양 R10	1060608	-271911904				
	T	^{ዘዘየ} R11	0	2				
🛿 Console 🛱 🤕 Tasks 🖹 Problems 📀 Executables 🔋 Memory			.	📰 🛃 📮 🔻 📬 🔻 🗖				
ectorAdd [C/C++ Application] gdb traces	nts" valu	="500"ll fi	le=" /src/vecto					
d.cu", fullname='/home/costroukhov/cuda-workspace/vectorAdd/src/vectorAdd(", line="36"}], lite="/src/vectorAd(", d.cu", fullname='/home/costroukhov/cuda-workspace/vectorAdd/src/vectorAdd.cu", line="36"}]								
470,340 (gdb)								
470 340 (odb)								
470,340 158°done,register-values=[{number="15",value="0"}]								



TOTALVIEW

- UNIX Symbolic Debugger for C/C++, Fortran, mixed Python/C++, PGI HPF, assembler programs
- JSC's "standard" debugger
- Advanced features
 - Multi-process and multi-threaded
 - Multi-dimensional array data visualization
 - Support for parallel debugging (MPI: automatic attach, message queues, OpenMP, Pthreads)
 - Scripting and batch debugging
 - Advanced memory debugging
 - Reverse debugging
 - CUDA and OpenACC support
 - Remote debugging
- NOTE: JSC license limited to 2048 processes (shared between all users)





Forschungszentrum

LINARO FORGE - DDT

- UNIX Graphical Debugger for C/C++, Fortran, and Python programs
- Modern, easy-to-use debugger
- Advanced features
 - Multi-process and multi-threaded
 - Multi-dimesional array data visualization
 - Support for MPI parallel debugging (automatic attach, message queues)
 - Support for OpenMP (Version 2.x and later)
 - Support for CUDA and OpenACC
 - Job submission from within debugger
- https://linaroforge.com/linaroDdt
- NOTE: JSC license limited to 128 processes (shared between all users)





Mitglied der Helmholtz-Gemeinschaft

INTEL INSPECTOR

- Detects memory and threading errors
 - Memory leaks, corruption and illegal accesses
 - Data races and deadlocks
- Dynamic instrumentation requiring no recompilation
- Supports C/C++ and Fortran as well as third party libraries
- Multi-level analysis to adjust overhead and analysis capabilities
- API to limit analysis range to eliminate false positives and speed-up analysis





INTEL INSPECTOR: GUI

r000ti2	×										
<u>70</u> c	Detect	t Deadlocks and Data Races					INTEL I	ISPECTOR			
0	Target	📻 Analysis Type 🕜 Collection Log 🛛 🥹 Summary						11 .			
Probles	ns					5	Filters	Sort * * 😵			
D .	•	Type Sources	Modules		State		Severity	<u>^</u>			
EP1	0	Data race find_and_fix_threading_errors.cpp; task_scheduler_init.h	find_and_fix_the	reading_errors.	exe Pa New		Error	3 item(s)			
EP2	0	Data race blocked_range.h; parallel_for.h; partitioner.h; task.h	find_and_fix_the	reading_errors.4	exe Pa New		Туре				
EP3	0	Data race winvideo.h	find_and_fix_the	reading_errors.	exe Pe New		Data race	3 item(s)			
							Source				
							blocked_range.h	1 item(s)			
							find_and_fix_threading_errors.c	1 item(s)			
							parallel_tor.h	1 item(s)			
							task.h	Titem(s)			
							task scheduler init.h	1 item(s)			
							winvideo.h	1 item(s)			
							Hodule				
_							find and fix threading errors er	e Litemis) 🗡			
9.1		1 of 12 P Al Code Locations: Data	ace	/////		😨 Tim	eline	Ŵ			
Descript	ion	Source Fun	ction	Module		Variable					
103	6	primary.scene = 4scene;	find a	and fix thre	ading error	s.exeirender o	read_video (2108)				
104									Look of L Look and Look	VE 20	1 -
105		col=trace(aprimary); //Threading Error: col is a :	plobs		Detect	Memory Probl	ems		Intel Inspect	OF XE 20	15
101		// 1) Make col a local variable									
Write		find_and_fix_threading_errors.cpp:105_rend	ler_one_pixel 1	. < 🔴	Target	Analysis Type	💊 Collection Log	Summary			Þ
103	19 C	primary.scene = sscene;	find_s								0
104		and an and an and a state of the state of th	find_e	Proble	ms						R.
100		//2 ways to fix this threading error	1100-0	ID	G h	Type		Sources		State	
101	12	<pre>// 1) Make col a local variable</pre>		(T) P1	0			Cod and Com		D. Maria	-
HINT:	Synchro	onization allocation site_task_scheduler_init.h:104 task	scheduler_init f	6 CPI		Mismatched a	llocation/deallocation	rind_and_fix_n	nemory_errors	Pe New	
7102	13	thread_stack_size = TBB_USE_CAPTURED_EXCEPTIO	S 7 p find_s	e e P2	8	Invalid memor	ry access	find_and_fix_n	nemory_errors	New New	
				• P3		Memory grow	th	[Unknown]; fi	nd_and_fix_me	P Not fixed	
				● P4	A	Memory growt	th	Unknown]: fi	nd and fix me	₽ Confirmer	4
				_				(_
				< 1 0	È.		1 of 2 ▷ All Co	de Locations:	Invalid memory a	access	8
				Descrip	tion Sou	urce	Function Mod	ule	Object	t Size Offset	
				Mitte	fin	d and fiv memory	anerstor find	and fir mame			
				write	1100	_and_nx_memory_e	enor operator() find_	and_nx_memo	ny_enor		_
				16	4	ALCONT OF MANY PROPERTY.		find_	and_fix_memory	y_errors.ex	ce.
				16	5	for (unsigne	ed int i=0;i<=(mbox	size find_	and_fix_memory	y_errors.ex	(e
				16	6	local_mk	<pre>pox[i]=0; //Memory</pre>	Err find_	and_fix_memory	y_errors.ex	ke
				16	7			find	and fix memory	y_errors.ex	ce
				16	8	for (int y =	- r.begin(); y != r	end thb d	ebug.dll!loca!	1_wait for	a

Forschungszentrum | CENTRE



- Data race detector for large OpenMP programs
- Combination of static and dynamic techniques
 - Low runtime and memory overhead
 - Still high accuracy and precision
- Now part of LLVM
- Compile with -fsanitize=thread
- Can be used with GCC, but CLANG OpenMP runtime must be linked
- Creates output in text format



ARCHER EXAMPLE

RNING: ThreadSanitizer: data race (pid=2234)

Vrite of size 4 at 0x7fff81d209d0 by thread T1

- #0 .omp_outlined._debug_ /p/project/training2410/knobloch1/archer_test.c:11:10 (a.out+0xd1efb)
- #1 .omp_outlined. /p/project/training2410/knobloch1/archer_test.c:9:1 (a.out+0xd1f85)
- #2 __kmp_invoke_microtask <null> (libomp.so+0xb8782)
- #3 main /p/project/training2410/knobloch1/archer_test.c:9:1 (a.out+0xd1d55)

Previous read of size 4 at 0x7fff81d209d0 by main thread:

- #0 .omp_outlined._debug_ /p/project/training2410/knobloch1/archer_test.c:11:12 (a.out+0xd1ed6)
- #1 .omp_outlined. /p/project/training2410/knobloch1/archer_test.c:9:1 (a.out+0xd1f85)
- #2 __kmp_invoke_microtask <null> (libomp.so+0xb8782)
- #3 main /p/project/training2410/knobloch1/archer_test.c:9:1 (a.out+0xd1d55)

Location is stack of main thread.

Location is global '??' at 0x7fff81d03000 ([stack]+0x1d9d0)

Thread T1 (tid=2237, running) created by main thread at:

#0 pthread_create /dev/shm/swmanage/jusuf/Clang/16.0.6/GCCcore-12.3.0/llvm-project-16.0.6.src/compiler-rt/lib/tsan/rtl/tsan_interceptors_posix.cpp:1048:3 (a.out+0x2678b)
#1 __kmp_create_worker <null> (libomp.so+0x97676)

SUMMARY: ThreadSanitizer: data race /p/project/training2410/knobloch1/archer_test.c:11:10 in .omp_outlined._debug__

JÜLICH JÜLICH SUPERCOMPUTING CENTRE

MUST

- Next generation MPI correctness and portability checker
- https://www.i12.rwth-aachen.de/go/id/nrbe
- MUST reports
 - Errors: violations of the MPI-standard
 - Warnings: unusual behavior or possible problems
 - Notes: harmless but remarkable behavior
 - Potential deadlock detection
- Usage
 - Compile with debug information (i.e. use the -g flag)
 - Run application under the control of mustrun (requires (at least) one additional MPI process)
 - E.g. on JUSUF: mustrun --must:mpiexec srun --must:np -n -n 4 ./app
 - Open output html report (might need to copy it to your local machine)



MUST DATATYPE MISMATCH

Rank	Туре	Message	From	References
0	Error	A send and a receive operation use datatypes that do not match! Mismatch occurs at (contiguous) [0](MPI_INT) in the send type and at (MPI_BYTE) in the receive type (consult the MUST manual for a detailed description of datatype positions). A graphical representation of this situation is available in a <u>detailed type mismatch view (MUST_Output-files/MUST_Typemismatch_0.html</u>). The send operation was started at reference 1, the receive operation was started at reference 2. (Information on communicator: MPI_COMM_WORLD) (Information on send of count 1 with type:Datatype created at reference 3 is for C, commited at reference 4, based on the following type(s): { MPI_INT}Typemap = {(MPI_INT, 0), (MPI_INT, 4)}) (Information on receive of count 8 with type:MPI_BYTE)	MPI_Sendrecv called from: #0 main@example.c:33	reference 1 rank 0: MPI_Sendrecv called from: #0 main@example.c:33 reference 2 rank 1: MPI_Sendrecv called from: #0 main@example.c:33 reference 3 rank 0: MPI_Type_contiguous called from: #0 main@example.c:29 reference 4 rank 0: MPI_Type_commit called from: #0 main@example.c:30



MUST DEADLOCK DETECTION





DEBUGGING RECOMMENDATIONS

- Always debug at the lowest possible scale!
- GPU Applications:
 - Single Node / Workstation: Use CUDA-GDB
 - Multi-Node / Supercomputer: Use TotalView/DDT
- MPI Applications:
 - Check with MUST at least once
 - Use TotalView/DDT at small scale (if error occurs there), else attach to as few processes as neccessary



DON'T BE THESE GUYS







JÜLICH SUPERCOMPUTING CENTRE

REMINDER: DEBUGGING CAN BE FRUSTRATING







PERFORMANCE ANALYSIS TOOLS



TODAY: THE "FREE LUNCH" IS OVER

- Moore's law is still in charge, but
 - Clock rates no longer increase
 - Performance gains only through increased parallelism
- Optimization of applications more difficult
 - Increasing application complexity
 - Multi-physics
 - Multi-scale
 - Increasing machine complexity
 - Hierarchical networks / memory
 - Many-core CPUs and Accelerators
 - Modular Supercomputing Architecture
- Every doubling of scale reveals a new bottleneck!



48 Years of Microprocessor Trend Data

New plot and data collected for 2010-2019 by K. Rupp



PERFORMANCE FACTORS

- "Sequential" (single core) factors
 - Computation
 - Thoose right algorithm, use optimizing compiler
 - Vectorization
 - Choose right algorithm, use optimizing compiler
 - Cache and memory
 - Choose the right data structures and data layout



PERFORMANCE FACTORS

- "Parallel" (multi core/node) factors
 - Partitioning / decomposition

Load balancing

- Communication (i.e., message passing)
- Multithreading
- Core binding / NUMA
- Synchronization / locking
- I/O
 - Often not given enough attention
 - Parallel I/O matters



TUNING BASICS

- Carefully set various tuning parameters
 - The right (parallel) algorithms and libraries
 - Compiler flags and directives
 - Correct machine usage (mapping and bindings)

Get the most performance before tuning!

- Measurement is better than guessing
 - To determine performance bottlenecks
 - To compare alternatives
 - To validate tuning decisions and optimizations
 - After each step!

PERFORMANCE ENGINEERING WORKFLOW



- Prepare application (with symbols), insert extra code (probes/hooks)
- Collection of data relevant to execution performance analysis
- Calculation of metrics, identification of performance metrics
- Presentation of results in an intuitive/understandable form
- Modifications intended to eliminate/reduce performance problems



THE 80/20 RULE

- Programs typically spend 80% of their time in 20% of the code
 - Show what matters!
- Developers typically spend 20% of their effort to get 80% of the total speedup possible for the application

F Know when to stop!

Don't optimize what does not matter
 ^T Make the common case fast!



PERFORMANCE MEASUREMENT

Two dimensions

When performance measurement is triggered

- External trigger (asynchronous)
 - Sampling
 - Trigger: Timer interrupt OR Hardware counters overflow
- Internal trigger (synchronous)
 - Code instrumentation (automatic or manual)

How performance data is recorded

• Profile

• Summation of events over time

• Trace

• Sequence of events over time



MEASUREMENT METHODS: PROFILING

- Recording of aggregated information
 - Time
 - Counts
 - Calls
 - Hardware counters
- Across program and system entities
 - Functions, call sites, loops, basic blocks, ...
 - Processes, threads
- Statistical information
 - Min, max, mean and total number of values

Advantages

+ Works also for long-running programs

Disadvantages

 Variations over time get lost



PROFILING: ISSUES RELATED TO "AVERAGING"

• Moving bottleneck across processors can "average out" imbalances



Imbalance changes over time ⇒ problem might not appear in short runs!



MEASUREMENT METHODS: TRACING

- Recording information about significant points (events) during execution of the program
 - Enter/leave a code region (function, loop, ...)
 - Send/receive a message ...
- Save information in event record
 - Timestamp, location ID, event type
 - plus event specific information
- Event trace := stream of event records sorted by time

Advantages

- + Can be used to reconstruct the dynamic behavior
- + Profiles can be calculated out of trace data

Disadvantages

- HUGE trace files
- Can only be used for short durations or small configurations
 - JÜLICH SUPERCOMPUTING CENTRE

⇒ Abstract execution model on level of defined events


EVENT TRACING: "TIMELINE" VISUALIZATION





CRITICAL ISSUES

- Accuracy
 - Intrusion overhead
 - Measurement takes time and thus lowers performance
 - Perturbation
 - Measurement alters program behaviour
 - E.g., memory access pattern
 - Accuracy of timers & counters
- Granularity
 - How many measurements?
 - How much information / processing during each measurement?
- Tradeoff: Accuracy vs. Expressiveness of data



REMARK: NO SINGLE SOLUTION IS SUFFICIENT!



A combination of different methods, tools and techniques is typically needed!

- Analysis
 - Statistics, visualization, automatic analysis, data mining, ...
- Measurement
 - Sampling / instrumentation, profiling / tracing, ...
- Instrumentation
 - Source code / binary, manual / automatic, ...



PERFORMANCE TOOLS (STATUS: NOV 2024)

- Score-P
- Scalasca
- Vampir[Server]
- Linaro Forge
 - Performance Reports
 - MAP
- Intel Tools
 - VTune Amplifier XE
 - Intel Advisor
- AMD uProf
- NVIDIA Tools
 - Nsight Systems
 - Nsight Compute
- Darshan

• ...

Mitglied der Helmholtz-Gemeinschaft







- Community-developed
 open-source
- Replaced tool-specific instrumentation and measurement components of partners
- <u>http://www.score-p.org</u>









UNIVERSITY OF OREGON







Score-P FUNCTIONALITY

- Provide typical functionality for HPC performance tools
- Instrumentation (various methods)
 - Multi-process paradigms (MPI, SHMEM)
 - Thread-parallel paradigms (OpenMP, POSIX threads)
 - Accelerator-based paradigms (OpenACC, CUDA, OpenCL. Kokkos)
 - In any combination!
- Flexible **measurement** without re-compilation:
 - Basic and advanced **profile** generation (⇒ CUBE4 format)
 - Event **trace** recording (⇔ OTF2 format)
- Highly scalable I/O functionality
- Support all fundamental concepts of partner's tools





CUBE EXAMPLE



Selected "!\$omp do @z_solve.prep.f.52"

Mitglied der Helmholtz-Gemeinschaft

Forschungszentrum CENTRE

•

SCORE-P: ADVANCED FEATURES

- Measurement can be extensively configured via
 environment variables
 ONE DOES NOT SIMPLY
- Allows for targeted measurements:
 - Selective recording
 - Phase profiling
 - Parameter-based profiling

- FIX PERFORMANCE ISSUES
- GPU support: CUDA, OpenACC, OpenCL, HIP, Kokkos, ...
- Please ask us or see the user manual for details



SCALASCA



http://www.scalasca.org/

- Scalable Analysis of Large Scale Applications
- Approach
 - Instrument C, C++, and Fortran parallel applications (with Score-P)
 - Option 1: scalable call-path profiling
 - Option 2: scalable event trace analysis
 - Collect event traces
 - Process trace in parallel
 - Wait-state analysis
 - Delay and root-cause analysis
 - Critical path analysis
 - Categorize and rank results





AUTOMATIC TRACE ANALYSIS

- Automatic search for patterns of inefficient behaviour
- Classification of behaviour & quantification of significance
- Identification of delays as root causes of inefficiencies





- Guaranteed to cover the entire event trace
- Quicker than manual/visual trace analysis
- Parallel replay analysis exploits available memory & processors to deliver scalability



EXAMPLE MPI WAIT STATES





SCALASCA ROOT CAUSE ANALYSIS

Root-cause analysis

- Wait states typically caused by load or communication imbalances earlier in the program
- Waiting time can also propagate (e.g., indirect waiting time)
- Enhanced performance analysis to find the root cause of wait states

Approach

- Distinguish between direct and indirect waiting time
- Identify call path/process combinations delaying other processes and causing first order waiting time
- Identify original delay



SCALASCA TRACE ANALYSIS EXAMPLE





VAMPIR EVENT TRACE VISUALIZER

- Offline trace visualization for Score-Ps OTF2 trace files
- Visualization of MPI, OpenMP and application events:
 - All diagrams highly customizable (through context menus)
 - Large variety of displays for ANY part of the trace
- http://www.vampir.eu
- Advantage:
 - Detailed view of dynamic application behavior
- Disadvantage:
 - Completely manual analysis
 - Too many details can hide the relevant parts





EVENT TRACE VISUALIZATION WITH VAMPIR

- Visualization of dynamic runtime behaviour at any level of detail along with statistics and performance metrics
- Alternative and supplement to automatic analysis
- Typical questions that Vampir helps to answer
 - What happens in my application execution during a given time in a given process or thread?
 - How do the communication patterns of my application execute on a real system?
 - Are there any imbalances in computation, I/O or memory usage and how do they affect the parallel execution of my application?

Timeline charts

 Application activities and communication along a time axis



Summary charts

 Quantitative results for the currently selected time interval



CENTRE

VAMPIR PERFORMANCE CHARTS

Timeline Charts



- Master Timeline
- Process Timeline
- Summary Timeline
- Performance Radar
- Counter Data Timeline
 - I/O Timeline

all threads' activities

- single thread's activities
- all threads' function call statistics
- all threads' performance metrics
- single threads' performance metrics
- all threads' I/O activities

Summary Charts



Function Summary



- Message Summary
- I/O Summary



Process Summary







VAMPIR DISPLAYS





LINARO PERFORMANCE REPORTS



- Single page report provides quick overview of performance issues
- Works on unmodified, optimized executables
- Shows CPU, memory, network and I/O utilization
- Supports MPI, multi-threading and accelerators
- Saves data in HTML, CVS or text form
- <u>https://www.linaroforge.com/linaroPerformanceReports</u>
- Note: License limited to 128 processes (with unlimited number of threads)



EXAMPLE PERFORMANCE REPORTS

Summary: cp2k.popt is CPU-bound in this configuration

The total wallclock time was spent as follows:



Time spent running application code. High values are usually good. This is average; check the CPU performance section for optimization advice. Time spent in MPI calls. High values are usually bad. This is average; check the MPI breakdown for advice on reducing it. Time spent in filesystem I/O. High values are usually bad.

This is **negligible**; there's no need to investigate I/O performance.

This application run was CPU-bound. A breakdown of this time and advice for investigating further is in the CPU section below.

CPU

A breakdown of how the 56.5% total CPU time was spent:

Scalar numeric ops	27.7%	
Vector numeric ops	11.3%	
Memory accesses	60.9%	
Other	0.0	1

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.

Little time is spent in vectorized instructions. Check the compiler's vectorization advice to see why key loops could not be vectorized.

I/O

A breakdown of how the 0.0% total I/O time was spent:

Time in reads	0.0%
Time in writes	0.0%

Estimated read rate 0 bytes/s

Estimated write rate 0 bytes/s

No time is spent in I/O operations. There's nothing to optimize here!

MPI

Of the 43.5% total time spent in MPI calls: Time in collective colle 0 00/

Time in conective cans	0.270	
Time in point-to-point calls	91.8%	
Estimated collective rate	169 Mb/s	
Estimated point-to-point rate	50.6 Mb/s	

The point-to-point transfer rate is low. This can be caused by inefficient message sizes, such as many small messages, or by imbalanced workloads causing processes to wait. Use an MPI profiler to identify the problematic calls and ranks.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage 82.5 Mb Peak process memory usage 89.3 Mb Peak node memory usage 7.4%

The peak node memory usage is low. You may be able to reduce the total number of CPU hours used by running with fewer MPI processes and more data on each process.



NVIDIA TOOLS -- LEGACY TRANSITION





NSIGHT SYSTEM

- System-wide application tuning
- Locate optimization opportunities
 - Visualize millions of events on a timeline
 - See gaps of unused CPU and GPU time
- Balance workloads across multiple CPUs and GPUs
 - CPU utilization and thread state
 - GPU streams, kernels, memory transfers, etc.
- Multi-platform support
 - Linux, Windows and Mac OS X (host-only)
- x86-64, Power9, ARM server, Tegra (Linux & QNX)





GPU METRIC SAMPLING

	 19.66 16.78 16.88 	10.93 1/8	17.18 17.29 17.29 17.48	17.35 17.58 17.76 17.88 17.95
 CPU (256) 				
 GPU (A100 Graphics Device - GPU Metrics 	the last state of the second s	Industry maline	Is my CDU full? Sufficient	gride size & streams?
GPC Clock Frequency		•	is my GPO rull: Sufficient	grius size a screams:
SYS Clock Frequency GR Active	Land Land Market Market		Is my instruction rate low	(possibly IO bound)?
SM Active	and the second support the second	and collision decision	is my moer decion race ton	(possibly to bound).
SM Instructions	Allenania Allenania and Anno Anno Anno Anno Anno Anno Anno An	MINUT 214 TIA. HI LI	Am I using tensor cores?	
SM Warp Occupancy	Inter States		Can I see GPU Direct RDM	A/Storage or other transfers?
DRAM Bandwidth	he d as the he has a	kinitin kin	System wide CDU absorve	tion
 PCIe Bandwidth GPU (A100 Graphics Device - 	- 00	•	System-wide GPU observa	
* GPU Metrics	needlessee material material	had been a been a	(no app required, but sudo or reg	key)
GPC Clock Frequency			10kHz default can be incr	cosed depending on CDU
GR Active	ALANA DATA DATA DATA DATA DATA DATA DATA D	-	TUKITZ GETAULL CALL DE INCL	eased depending on GPU
SM Active	the second s		Metrics	
SM Instructions			meenes.	
SM Warp Occupancy	aka da aka an tang ang aka ka			
b DDAM Bacdwidth	all a share a s		SM utilizations	IO throughputs
 PCIe Bandwidth 	$(k_1, d_1) = k_1 + 1$, $(k_1, d_2) = k_1 + 1$, $(k_1, d_2) = k_1 + 1$, $(k_1, d_2) = k_1 + 1$.	CE DIDIO LAND	Citie antitice	DCIa
 CPU (A100 Graphics Device - 	- OC		 SMs active 	• PCIe
 GPU Metrics GPU (A100 Graphics Device - 	ot	thereis in here to the	Instructions	 NVI ink
GPU Metrics	The second secon	La Lister for suite		
 GPU (A100 Graphics Device - GPU Metrics 	- OC	hard structure in the	 TensorCores 	 DRAM
 GPU (A100 Graphics Device - 		A MALINAR, UNIVERSITE AND ADD.	- Warp accupancy	
 GPU Metrics GPU (A100 Graphics Device - 	- 0(and a strength to be a strength of the	• warp occupancy	
GPU Metrics	man in the localization in the surface and		(including unallocate	d slots)
* GPU (A100 Graphics Device -	- OC		(including unallocate	.u 5(0(5))
CUDA HW (Unknown GPU)		100110	anal bara bayon blead	(speak) grouped street haven) aparticle grouped street, haven)
2 [29106] python	T NYO IN COMMAND			
CUDA API		netAnyn Alfalaid	nEverEgnizonia	
✓ [24832] pythun •	x		le le	nsorFlow on 8xGA100 at 20kHz
CUDA API	F	a secondar of a first	astearsystema	



MULTI NODE SUPPORT – SHMEM, MPI, UCX, AND NCCL

OS runtime libraries	971 px: 0 msec						
SHMEM	971 px, onset (shmem_float_p [7,145)	shme shmem	n_barrier_all [18,019 µs]	shmem_finalize [1,279 s]			
Profiler overhead	971 px; 0 msec	Begins: 0,472616s		Name and Article a			
430917]	971 px; 0 msec	Ends: 0,4726195 (+2,470 µs) Thread: 1430905					
174635] MPI Rank 0	• <u>1998</u>						
PI	MPI_Isend [7,319 µs]	MPI_Irecv [7,585 µs]		MPI_Waitall [11,493 µs]			
x	ucp_tag_send_nbx UCP tr	ucp_tag_recv_nbx UCP tran	isf	ucp_rkey			
	ucp_tag_send_nbx	urn tan send nhy LUCP	transfer processing [21.1	83 ucl	452 15 521,608/ms 1,62115 4521,640	ns +321,00ms +321,0	4000 +021,/05 +021,/205 +021,/405
tart & End	1019 px; 0 msec Ends: 0,177988s (+3,990 µs)		ucp_tag_re	cv_nbx UCP transfer processing [14,243 µs]	clAllReduce [26.3		
4.000 million (1997)	Thread: 174635		A		mema int sum r		nyshmern free [91,979 us
ler overhead	Categoryld: 2						the second
niler overhead eads hidden — -	Categoryid: 2 Category: UCP transfer submit		1			hoonize (cud	nvshmem_quiet [
ler overhead ads hidden – – Co	ompletion tracking of no	l on-blocking UC	P commun - CUDA HW (0 - [All Stream	nication operations	ncciKernel_AliReduce_RING_LL	hronize cud	(nvstimem_quiet [)
ads hidden – –	mpletion tracking of no	l on-blocking UC	P commun CUDA HW (0 F [All Stream F 79.5% Defa	Dication operations	ncciKernel_AliReduce_RING_LL	hronize cud)	(nvshmem_quiet [) (cuStreamSy) (barr) cuStr (barr) (barr)
ads hidden – –	ompletion tracking of no	on-blocking UC	P commun - CUDA HW (0 - [All Stream - 79.5% Defa 7.335 ^k 19.8% Stre	100:06:00.0 - 1 1063 px 1 msec sj 1063 px 1 msec barro an ult stream 7 1063 px 0 msec am 22 1063 px 0 msec barro	nccikemel AliReduce RING LL nccikernel AliReduce RING LL nccikernel AliReduce RING LL Begins: 1,521615 Ends: 1,521655 (+41,504 µs	hronize cud S NG_LL_Sum_int32_t s)	(nvshmem_quiet [) () cuStreamSy) (barr) cuStr (barr) (barr)
hier overhead eads hidden – –	ompletion tracking of no	l on-blocking UC	P commun - CUDA HW (0 - [All Stream - 79.5% Defa 7.335* 19.8% Stre	D00:06:00.0 - 1 1063 pro msec s] 1063 pro msec ult stream 7 am 22 1063 pro msec Ima pro msec Ima pro msec Ima pro msec Ima pro msec	nccikernel_AliReduce_RING_LL nccikernel_AliReduce_RING_LL nccikernel_AliReduce_RING_LL Begins: 1,521615 Ends: 1,521655 (+41,504 µs grid: <<<1, 1, 1>>> block: <<<64 + 1>>>	hronize cud. Solon NG_LL_Sum_int32_t s)	(nvshmem_quiet [) () cuStreamSy) barr,) cuStr barri barri barri barri
ads hidden – –	ompletion tracking of no	on-blocking UC	P commun - CUDA HW (0 - [All Stream - 79.5% Defa 7.335* 19.8% Stre Events View	s] 1063 pro msec s] 1063 pro	ncclKernel_AllReduce_RING_LL ncclKe	hronize cud. S NG_LL_Sum_int32_t s) 4722 bytes 0 bytes	(nvshmem_quiet [) () cuStreamSy) barr) cuStr barr barr barr barr barr
ads hidden – –	ompletion tracking of no	on-blocking UC	P commun - CUDA HW (0 - [All Stream - 79.5% Defa 7.335* 19.8% Stre Events View	ame	ncclKernel_AliReduce_RING_LL ncclKernel_AliReduce_RING_LL ncclKernel_AliReduce_RING_LL Begins: 1,52165s (+41,504 µs grid: <<<1, 1,1>>> block: <<<<4, 1, 1>>> block: <<<<64, 1, 1>>> Launch Type: Regular Static Shared Memory: 41. Dynamic Shared Memory: 41. Dynamic Shared Memory: 41. Bynamic Shared Mem	hronize cud	Name Description:
ads hidden – –	mpletion tracking of no	on-blocking UC	P commun - CUDA HW (0 - [All Stream - 79.5% Defa 7.335* 19.8% Stre Events View	ame Sincation operations D00:06:00.0 - 1 1063 prof msec I 1063 prof msec I 1063 prof msec I 1063 prof msec I 1063 prof msec I 1063 prof msec I 1063 prof msec I 1063 prof msec I 1063 prof msec I I I I I I I I I I I I I I I I I I I	ncclKernel_AllReduce_RING_LL ncclKernel_AllReduce_RING_LL ncclKernel_AllReduce_RING_LL Begins: 1,52161s Ends: 1,52165 (+41,504 µs grid: <<<1, 1, 1>>> block: <<<64, 1, 1>>> block: <<<64, 1, 1>>> Launch Type: Regular Static Shared Memory: 41, Dynamic Shared Memory: 41, Dynamic Shared Memory: 41, Local Memory Total: 251.6	hronize cud Source for the second	Name Description: nyshmemx_int_sum_reduce_on
ads hidden – –	mpletion tracking of no	on-blocking UC	P commun - CUDA HW (0 - [All Stream - 79.5% Defa 7.335* 19.8% Stre Events View # • N 730 1731	Anication operations D00:06:00.0 - 1 1063 px msec s] 1063 px msec bar a am 22 1063 px msec bar am 22 1063 px msec bar am 22 1063 px msec bar am 24 1063 px msec bar am 25 1063 px msec bar am 20 1063 px msec bar am 21 1063 px msec bar	nccikernel AliReduce_RING_LL nccikernel AliReduce_RING_LL Begins: 1,52161s Ends: 1,52165s (+41,504 µs grid: <<<1, 1, 1>>> block: <<<64, 1, 1>>> Launch Type: Regular Static Shared Memory: 41. Dynamic Shared Memory: 41. Dynamic Shared Memory: 41. Dynamic Shared Memory: 41. Cocal Memory Per Thread: 96 Local Memory Total: 251.6 Shared Memory Reactions: Shared Memory Rea	hronize cud. Solution NG_LL_Sum_int32_t s) 472 bytes 0 bytes 58.240 bytes 98.304 bytes 48.8 1962 1962	Name
eads hidden –	mpletion tracking of no	on-blocking UC	P commun - CUDA HW (0 - [All Stream - 79.5% Defa 7.335* 19.8% Stre Events View # * N 730 731 732	ame	nccikernel_AliReduce_RING_LL nccike	hronize cud. Solutions NG_LL_Sum_int32_t s) 472 bytes 0 bytes 58.240 bytes 98.304 bytes 98.304 bytes 48 621	Name
reads hidden –	mpletion tracking of no	on-blocking UC	P commun - CUDA HW (0 - [All Stream - 79.5% Defa 7.335* 19.8% Stre Events View # * * N 730 1731 732 733	ame	ncclKernel_AliReduce_RING_LL ncclKernel_AliReduce_RING_LL ncclKernel_AliReduce_RING_LL ncclKernel_AliReduce_RING_LL ncclKernel_AliReduce_RING_LL ncclKernel_AliReduce_RING_LL ncclKernel_AliReduce_RING_LL grid: <<<1, 1, 1>>> block: <<<64, 1, 1>>> block: <<64, 1, 1>>> block: <<<64, 1, 1>>> block: < 64, 1 >> block: < <td>hronize cud. Solution cud. NG_LL_Sum_int32_t s) 472 bytes 0 bytes 58.240 bytes 98.304 bytes 98.304 bytes 1962 1962 1962 1962</td> <td>Name</td>	hronize cud. Solution cud. NG_LL_Sum_int32_t s) 472 bytes 0 bytes 58.240 bytes 98.304 bytes 98.304 bytes 1962 1962 1962 1962	Name



OPENMP



OMPT-capable OpenMP runtime required



EXPERT SYSTEM

									CUDA Async	: Memcpy with Pa	igeable Men
• <u>View Tools H</u> elp ofile_circe-n011_506451_0.qdrep.qdrep ×									CUDA Synch	nronous Memcpy nronous Memset	
Timeline View *								📼 Q, 1x	CUDA Synch	nronization APIs	
55 -		+676,05ms	+676,1ms	+676,150	-6	76,2ms	+676,25ms	+676,3ms	CUDA GPU S	Starvation	
CUDA API 🐺 cudaStrea	m5	ec eleme	n elem	el a	idaMemc	cudaMemcpyAsyr	nc cuda	0	CUDA GPU I	Low Utilization	
OS runtime libraries						pthread_cond_wait			VULKAN GP	U Starvation	
10 threads hidden + 1179 px;	D ms								VULKAN GP	U Low Utilization	
✓ CUDA HW (0000:34:00.0 - Te	mis	ec							0		
Fall Streams] 🐺 🕽 ncclAll	ms	ec	1 1								
60.8% Default stream 1 1179 pc	mis	ec.	1 1								
A 20 206 Stream 20	in mar										
0.791 35.270 Stream 29											
0.791 33-279 Sureally 23											1
2.791 39-270 Stream 29		Duration •	Start	Src Kind	Dst Kind	Bytes	PID	Device ID	Context ID	Stream ID	Settin
UDA Async Memcpy with Pageable Memory	•	Duration * 2,048 µs	Start 6,38792s	Src Kind Device	Dst Kind Pageable	Bytes 8 B	PID 75475	Device ID 0	Context ID	Stream ID	Settin API Name cudaMemcq
UDA Async Memcpy with Pageable Memory the following APIs use PAGEABLE memory which causes asynchronous CUDA memcpy	*	Duration - 2,048 µs 2,048 µs	Start 6,38792s 6,8334s	Src Kind Device Device	Dst Kind Pageable Pageable	Bytes 8 B 4 B	PID 75475 75475	Device ID 0 0	Context ID 1	Stream ID 7	Settin API Name cudaMemcp cudaMemcp
UDA Async Memcpy with Pageable Memory the following APIs use PAGEABLE memory which causes asynchronous CUDA memcpy operations to block and be executed watch concerning to block and be executed	*	Duration * 2,048 µs 2,048 µs 2,016 µs	Start 6,38792s 6,8334s 2,5394s	Src Kind Device Device Device	Dst Kind Pageable Pageable Pageable	Bytes 8 B 4 B 4 B	PID 75475 75475 75475	Device ID 0 0	Context ID 1 1	Stream ID 7 7 7	Settin API Name cudaMemcq cudaMemcq cudaMemcq
UDA Async Memcpy with Pageable Memory the following APIs use PAGEABLE memory which causes asynchronous CUDA memcpy operations to block and be executed ynchronously. This leads to low GPU tillization.	•	Duration * 2,048 µs 2,048 µs 2,016 µs 2,016 µs	Start 6,38792s 6,8334s 2,5394s 3,90617s	Src Kind Device Device Device Device	Dst Kind Pageable Pageable Pageable Pageable	Bytes 8 B 4 B 4 B 4 B 4 B	PID 75475 75475 75475 75475 75475	Device ID 0 0 0 0	Context ID 1 1 1 1	Stream ID 7 7 7 7	Settin API Name cudaMemcp cudaMemcp cudaMemcp cudaMemcp
2.791 39.290 Stream 29 A A A A A A A A A A A A A	*	Duration 2,048 µs 2,048 µs 2,016 µs 2,016 µs 2,016 µs 2,016 µs	Start 6,38792s 6,8334s 2,5394s 3,90617s 4,25257s	Src Kind Device Device Device Device Device	Dst Kind Pageable Pageable Pageable Pageable Pageable	Bytes 8 B 4 B 4 B 48 B 48 B 48 B	PID 75475 75475 75475 75475 75475 75475	Device ID 0 0 0 0 0	Context ID 1 1 1 1 1	Stream ID 7 7 7 7 7 7	Settin API Name cudaMemcg cudaMemcg cudaMemcg cudaMemcg cudaMemcg cudaMemcg
UDA Async Memcpy with Pageable Memory the following APIs use PAGEABLE memory which causes asynchronous CUDA memcpy operations to block and be executed ynchronously. This leads to low GPU tilization.	*	Duration 2,048 µs 2,048 µs 2,016 µs 2,016 µs 2,016 µs 2,016 µs 2,016 µs	Start 6,38792s 6,8334s 2,5394s 3,90617s 4,25257s 5,67617s	Src Kind Device Device Device Device Device Device	Dst Kind Pageable Pageable Pageable Pageable Pageable Pageable	Bytes 8 B 4 B 4 B 48 B 48 B 48 B	PID 75475 75475 75475 75475 75475 75475 75475 75475	Device ID 0 0 0 0 0	Context ID 1 1 1 1 1	Stream ID 7 7 7 7 7 7 7	Settin API Name cudaMemcq cudaMemcq cudaMemcq cudaMemcq cudaMemcq cudaMemcq
CUDA Async Memcpy with Pageable Memory the following APIs use PAGEABLE memory which causes asynchronous CUDA memcpy sperations to block and be executed ynchronously. This leads to low GPU stilization. Suggestion: If applicable, use PINNED nemory instead. LI command:	*	Duration 2,048 µs 2,048 µs 2,016 µs 2,016 µs 2,016 µs 2,016 µs 2,016 µs 2,016 µs	Start 6,38792s 6,8334s 2,5394s 3,90617s 4,25257s 5,67617s 5,9572s	Src Kind Device Device Device Device Device Device Device Device	Dst Kind Pageable Pageable Pageable Pageable Pageable Pageable Pageable	Bytes 8 B 4 B 4 B 4 8 4 B 4 B 4 B 4 B 8 B	PID 75475 75475 75475 75475 75475 75475 75475 75475	Device ID 0 0 0 0 0 0 0 0 0	Context ID 1 1 1 1 1 1 1 1	Stream ID 7 7 7 7 7 7 7 7 7 7 7	Settin API Name cudaMemcg cudaMemcg cudaMemcg cudaMemcg cudaMemcg cudaMemcg cudaMemcg cudaMemcg cudaMemcg
Expert System View * CUDA Async Memcpy with Pageable Memory The following APIs use PAGEABLE memory which causes asynchronous CUDA memcpy operations to block and be executed synchronously. This leads to low GPU utilization. Suggestion: If applicable, use PINNED nemory instead. LI command: sys analyze -r cuda-async-memcpy /mnt/data asys analyze -r cuda-async-memcpy /mnt/data	1 • • •	Duration 2,048 µs 2,048 µs 2,016 µs 2,016 µs 2,016 µs 2,016 µs 2,016 µs 2,016 µs 2,016 µs	Start 6,38792s 6,8334s 2,5394s 3,90617s 4,25257s 5,67617s 5,9572s 5,9708s	Src Kind Device Device Device Device Device Device Device Device	Dst Kind Pageable Pageable Pageable Pageable Pageable Pageable Pageable Pageable	Bytes 8 B 4 B 4 B 48 B 48 B 48 B 48 B 48 B 48	PID 75475 75475 75475 75475 75475 75475 75475 75475	Device ID 0 0 0 0 0 0 0 0 0	Context ID 1 1 1 1 1 1 1 1 1 1 1	Stream ID 7 7 7 7 7 7 7 7 7 7 7 7 7	Settin API Name cudaMemcg c



NSIGHT COMPUTE

- Interactive CUDA kernel profiler
- Targeted metric sections for various performance aspects
- Customizable data collection and presentation (tables, charts, ...)
- GUI and CLI
- Python-based API for guided analysis and post-processing
- Support for remote profiling across machines and platforms

	- Launch	o-45045-Gence_tea_eat	ppeg_aoi		Contraction Provide	in occupancy calc	ulator					Coopy do	
	Launch			Time	Cycles R	gs GPU		SM Fre	quency	CC Proce	ISS		
Current	43843 - devid	ce_tea_leaf_ppcg_solve_init	(126, 1001, 1)x	(32, 4, 1) 217.63 usec	cond 297,114 40	0 - NVIDIA GeFord	ce RTX 2080 T	1.36 cyc	le/nsecon	d 7.5 (15958	3] tea_leaf		
) 🔺 Time	API Call ID	Function Name	Demangled N	l Process	Device Name	Grid Size	Block Siz	e		ycies [cycle] D	uration [msecond] Compute T	hroughput [%] N
0 2021-Dec	43843	device_tea_leaf_ppcg_	device_te_	[15958] tea_leaf	NVIDIA GeFor	e_ 126, 1001,		32,	4, 1	297,114	0.22	77	.89
1 2021-Dec-1	43857	device_tea_leaf_ppcg_sol	device_tea_I	[15958] tea_leaf	NVIDIA GeForce	RT 126, 1001		32,		1,264,921	0.94	54	.78
2 2021-Dec-1	43860	device_tea_leaf_ppcg_sol	device_tea_I	[15958] tea_leaf	NVIDIA GeForce	RT 126, 1001		32,	4, 1	1,462,446	1.07	86	.22
3 2021-Dec-1	43863	device_tea_leaf_ppcg_sol_	device_tea_I	[15958] tea_leaf	NVIDIA GeForce	RT 126, 1001		32,		1,443,836	1.06	23	3.81
age: Details	- Launch:	0 - 43843 - device_tea_leaf,	ppcg_sol +	🖓 👻 Add Baseline	▼ Apply <u>R</u> ules	🗟 Occupancy Calcu	ulator					Copy as	Image
	Launch			Time	Cycles Re	gs GPU		SM Free	uency	CC Proces	55	A A	0
Current	43843 - devid	ce tea leaf ppcg solve init i	(126, 1001, 1)x	(32, 4, 1) 217,63 usec	ond 297,114 40	0 - NVIDIA GeFord	e RTX 2080 Ti	1.36 cvcl	e/nsecond	7.5 115958	tea_leaf		•
GPU Speed Of Li	ight Through	put											0
Compute (SM) Throug	ghput [%]				77.89	Duration [usecond]						21	7.63
Memory Throughput	[%]				45.03	Elapsed Cycles [cycle)					297	,114
L1/TEX Cache Throug	hput [%]				68.22	SM Active Cycles [cy	cle]					293,38	5.19
L2 Cache Throughput	t [%]				2.30	SM Frequency [cycle.	/nsecond]						1.36
DRAM Throughput [%]	1				0.12	DRAM Frequency [cy	cle/nsecond]						6.79
🔥 High Compu	ite Throughpu	t Compute is more heavi whether any computati	ly utilized than ion is redundar	Memory: Look at the and could be reduced	d or moved to look	Analysis report sectio up tables.	n to see what i	the compu	te pipeline	s are spending	their time doing. Also, consider		
A FP64/32 Util	lization Wo ana	ratio of peak float (fp32) to https://www.angload ilysis.	double (fp64) that this kernel	performance on this d is fp64 bound, conside	evice is 32.1. The H er using 32-bit pred	ernel achieved 0% of th ision floating point op	his device's fp3 erations to im	2 peak per prove its pe	formance eformanc	and 19% of its e. See the <mark>Kern</mark>	fp64 peak performance. If Com <u>Profiling Guide</u> for mode detai	ls on roofline	e
- Compute Workle	oad Analysis												0
Executed Ipc Elapsed	[inst/cycle]				1.25	SM Busy [%]						7	8.61
Executed Ipc Active [in	nst/cycle]				1.27	Issue Slots Busy [%]							1.65



PROFILER REPORT

Selected result

1

Metric values

1

Page: Details 👻 Launch: 0-43843	3 - device_tea_leaf_ppcg_sol 👻 🍸 👻 Add Baseli	ne 💌 Apply <u>R</u> ules 🔚 Occupancy Calculato	r i	Copy as Image 💌
Launch	Time	Cycles Regs GPU	SM Frequency CC Proces	s $\oplus \Theta \oplus$
Current 43843 - device_tea_lea	f_ppcg_solve_init (126, 1001, 1)x(32, 4, 1) 217.63 u	second 297,114 40 0 - NVIDIA GeForce RT	X 2080 Ti 1.36 cycle/nsecond 7.5 [15958]	tea_leaf
GPU Speed Of Light Throughput				
Compute (SM) Throughput [%]		77.89 Duration [usecond]		217.63
Memory Throughput [%]		45.03 Elapsed Cycles [cycle]		297,114
L1/TEX Cache Throughput [%]		68.22 SM Active Cycles [cycle]		293,385.19
L2 Cache Throughput [%]		2.30 SM Frequency [cycle/nset	cond]	1.36
DRAM Throughput [%]		0.12 DRAM Frequency [cycle/n	second]	6.79
A High Compute Throughput Comp wheth	oute is more heavily utilized than Memory: Look at the ner any computation is redundant and could be redu	e <u>Compute Workload Analysis</u> report section to ced or moved to look-up tables.	see what the compute pipelines are spending t	their time doing. Also, consider 💿
The ratio of p A FP64/32 Utilization Workload Ana analysis.	eak float (fp32) to double (fp64) performance on thi alysis determines that this kernel is fp64 bound, con _	s device is 32:1. The kernel achieved 0% of this d sider using 32-bit precision floating point operati	evice's fp32 peak performance and 19% of its ons to improve its performance. See the <u>Kerre</u>	ip64 peak performance. If <u>Compute</u> <u>I Profiling Guide</u> for mode details on roofline
- Compute Workload Analysis				Q
Executed Ipc Elapsed [Inst/cycle]		1.25 SM Busy [%]		78.61
Executed Ipc Active [inst/cycle]		1.27 Issue Slots Busy [%]		31.65
Issued Ipc Active [inst/cycle]		1.27		

Expandable Sections Expert Analysis (Rules)



DATA TRANSFER ANALYSIS

- Detailed memory workload analysis chart and tables
- Shows transferred data or throughputs
- Tooltips provide metric names, calculation formulas and detailed background info





BASELINE COMPARISON

- Comparison of results directly within the tool with "Baselines"
- Supported across kernels, reports, and GPU architectures

Current	655 - reduceFinal (1, 1, 1)x(512	, 1, 1) 5.98 usecond	4,807 16	NVIDIA GeFo	orce RTX 2080 Ti 8	01.61 cycle/usecond	7.5 [4969] simpleCuc	aGraphs		
Baseline	654 - reduce (512, 1, 1)x(512, 1	, 1) 234.72 usecond	318,783 16	NVIDIA GeF	orce RTX 2080 Ti 1	.35 cycle/nsecond	7.5 [4969] simpleCuc	aGraphs		
■ GPU Speed Of I	iaht Throughput							GPUT	aroughput Chart	- 0
Compute (SM) Thr	oughput [%]			0.33 (-99.58%)	Duration Jusecond	1		0.01	noughput on an	5 98 (-97 45%)
Memory Throughou	ut [%]			0.80 (-98.25%)	Flapsed Cycles for	rcle]				4 807 (-98 49%)
L1/TEX Cache Thro	oughput [%]		1	1.66 (+12.76%)	SM Active Cycles	cycle]				46.09 (-99.98%)
L2 Cache Through	put [%]			0.80 (-94.49%)	SM Frequency [cy	le/usecond]			8	01.61 (-40.80%)
DRAM Throughput	[%]			0.48 (-98.94%)	DRAM Frequency	cycle/nsecond]				3.98 (-41.14%)
🛕 Small Grid	This kernel grid is too small to fill t	he available resources on	this device, res	ulting in only 0.0	0 full waves across a	ll SMs. Look at <u>Laun</u> d	<u>h Statistics</u> for more de	tails.		
 Roofline An 	alysis The ratio of peak float (fp32) to double (fp64) perform	nance on this d	evice is 32:1. Th	ne kernel achieved 0	% of this device's fp32	peak performance and	close to 0% of i	ts fp64 peak pe	formance.
				GPU TI	hroughput					
Compute (SM) [%			<u> </u>							
			1					_		
Memory [%	6) <mark></mark>				- ,					
	0.0 10.0	20.0 3	0.0	40.0	50.0	60.0	70.0	80.0	90.0	100.0
				Sp	peed Of Light (SOL)	%]				
Compute Workle	oad Analysis									Q
Executed Ipc Elaps	ed [inst/cycle]			0.00 (-99.16%)	SM Busy [%]					34.72 (-59.93%)
Executed Ipc Active	e [inst/cycle]			0.26 (-20.53%)	Issue Slots Busy [6]				7.11 (-14.71%)
Issued Ipc Active [i	nst/cycle]			0.28 (-14.71%)						
 Balanced 	No pipeline is over-utilized.									
- Memory Worklo	ad Analysis									, ⊳
									~	
Memory Throughpu	ut [Gbyte/second]			1.84 (-99.38%)	Mem Busy [%]					0.80 (-94.49%)
Memory Throughpu L1/TEX Hit Rate [%]	ut [Gbyte/second]]			1.84 (-99.38%) 0 (-100.00%)	Mem Busy [%] Max Bandwidth [%]			<u>~"</u>	0.80 (-94.49%) 0.57 (-98.75%)
Memory Throughpu L1/TEX Hit Rate [%] L2 Hit Rate [%]	ut [Gbyte/second]]		71.25	1.84 (-99.38%) 0 (-100.00%) 5 (+43,797.06%)	Mem Busy [%] Max Bandwidth [% Mem Pipes Busy [] %]				0.80 (-94.49%) 0.57 (-98.75%) 0.11 (-99.15%)
Memory Throughpu L1/TEX Hit Rate [%] L2 Hit Rate [%]	ut [Gbyte/second] .]		71.25	1.84 (-99.38%) 0 (-100.00%) 5 (+43,797.06%) Memo	Mem Busy [%] Max Bandwidth [% Mem Pipes Busy [ory Chart] %]			<u> </u>	0.80 (-94.49%) 0.57 (-98.75%) 0.11 (-99.15%)
Memory Throughpu L1/TEX Hit Rate [%] L2 Hit Rate [%]	ut [Gbyte/second] .]	16	71.25 00 Rea	1.84 (-99.38%) 0 (-100.00%) 5 (+43,797.06%) Mem o	Mem Busy [%] Max Bandwidth [% Mem Pipes Busy [ory Chart] %]			<u>(All</u>	0.80 (-94.49%) 0.57 (-98.75%) 0.11 (-99.15%)
Memory Throughpu L1/TEX Hit Rate [%] L2 Hit Rate [%]	ut [Gbyte/second]] 17.00 Inst (100 00%)	16	71.25 00 Req 00.00%)	1.84 (-99.38%) 0 (-100.00%) 5 (+43,797.06%) Mem o	Mem Busy [%] Max Bandwidth [% Mem Pipes Busy [ory Chart] k]			~" 	0.80 (-94.49%) 0.57 (-98.75%) 0.11 (-99.15%)
Memory Throughpa L1/TEX Hit Rate [%] L2 Hit Rate [%]	17.00 Inst (-100.00%)	16 (-11	71.25 .00 Req 00.00%)	1.84 (-99.38%) 0 (-100.00%) 5 (+43,797.06%) Mem e	Mem Busy [%] Max Bandwidth [% Mem Pipes Busy [ory Chart] k]	0.00 B	o	ی. هو 100%	0.80 (-94.49%) 0.57 (-98.75%) 0.11 (-99.15%)
Memory Throughpi L1/TEX Hit Rate [%] L2 Hit Rate [%]	17.00 Inst (-100.00%)	16 (-1) Giobal	71.25 00 Req 00.00%) 00 Req	1.84 (-99.38%) 0 (-100.00%) 5 (+43,797.06%) Mem	Mem Busy [%] Max Bandwidth [% Mem Pipes Busy [ory Chart] &]	0.00 B (+0.00%)	temory	ж ас «	0.80 (-94.49%) 0.57 (-98.75%) 0.11 (-99.15%)
Memory Throughpu L1/TEX Hit Rate [%] L2 Hit Rate [%]	17.00 Inst (-100.00%)	16 Global	71.25 00 Req 00.00%) 00 Req 9.80%)	1.84 (-99.38%) 0 (-100.00%) 5 (+43,797.06%) Mem	Mem Busy [%] Max Bandwidth [% Mem Pipes Busy [ory Chart) k)	0.00 B (+0.00%)	m Memory	Х" Холого Холо Хол	0.80 (-94.49% 0.57 (-98.75% 0.11 (-99.15%)
Memory Throughpu L1/TEX Hit Rate [%] L2 Hit Rate [%]	17.00 inst (-100.00%)	16 Global 1. (-9 0. 4	71.25 00.00%) 00.00%) 00.Req 9.80%) 00.Req 0.00%)	1.84 (-99.38%) 0 (-100.00%) 5 (+43,797.06%) Mem	Mem Busy [‰] Max Bandwidth [% Mem Pipes Busy [ory Chart) k)	0.00 B (+0.00%)	stem Memory	ත් ^{100%} දී - 80%	0.80 (-94.49%) 0.57 (-98.75%) 0.11 (-99.15%)
Memory Throughp L1/TEX Hit Rate [%] L2 Hit Rate [%]	ut [Gbyte/second] ;] 17.00 Inst (-100.00%) 0.00 Inst (+0.00%)	16 Global 1. (-9 0. (+ Local	71.25 00.Req 00.00%) 00.Req 9.80%) 00.Req 0.00%)	1.84 (-99.38%) 0 (-100.00%) 5 (+43,797.06%) Mem	Mem Busy [%] Max Bandwidth [% Mem Pipes Busy [ory Chart	 k]	0.00 B (+0.00%) 0.00 B (+0.00%)	System Memory	₩ 80%	0.80 (-94.49%) 0.57 (-98.75%) 0.11 (-99.15%)
Memory Throughpi L1/TEX Hit Rate [%] L2 Hit Rate [%]	17.00 Inst (-100.00%)	16 Global 1. (-1 (-1) (-1) (-1) (-1) (-1) (-1) (-1)	71.25 00 Req 00.00%) 00 Req 9.80%) 00 Req 0.00%) 00 Req	1.84 (-99.38%) 0 (-100.00%) 5 (+43,797.06%) Memo L1/TEX Cacha	4.00 KB] k]	0.00 B (+0.00%) 0.00 B (+0.00%)	System Mernory	₩ 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8	0.80 (-94.49%) 0.57 (-98.75%) 0.11 (-99.15%)



ROOFLINE ANALYSIS

- Determine whether the application is memory bound or compute bound
- Guided analysis points to detailed analysis of the most severe problem





DARSHAN

- I/O characterization tool logging parallel application file access
- Summary report provides quick overview of performance issues
- Works on unmodified, optimized executables
- Shows counts of file access operations, times for key operations, histograms of accesses, etc.
- Supports POSIX, MPI-IO, HDF5, PnetCDF, ...
- Binary log file written at exit post-processed into PDF report
- http://www.mcs.anl.gov/research/projects/darshan/
- Open Source: installed on many HPC systems



EXAMPLE DARSHAN REPORT EXTRACT



Mitglied der Helmholtz-Gemeinschaft

PERFORMANCE ANALYSIS RECOMMENDATIONS

- Measure and analyze at the desired scale (once you have a reasonable measurement setup)
- Get performance overview with Performance Reports
 - CPU Issues:
 - Use MAP, Vtune (on Intel nodes), or uProf (on AMD nodes)
 - Use perf / LIKWID / PAPI
 - MPI Issues: Use Scalasca/Vampir
 - GPU Issues: Use NVIDIA tools
 - I/O Issues: Use DARSHAN
- OR: Do it all with Score-P/Scalasca/Vampir



NEED HELP?

- Talk to the experts
 - Use local 1st-level support, e.g. SC support, SimLab
 - Use mailing lists
 - JSC/NVIDIA Application Lab
 - ATML Parallel Performance
 - ATML Application Optimization and User Service Tools

© Successful performance engineering often is a collaborative effort




QUESTIONS



Mitglied der Helmholtz-Gemeinschaft