# **ROC-profiler and debugger:**An Overview of AMD $ROCm^{TM}$ Tools

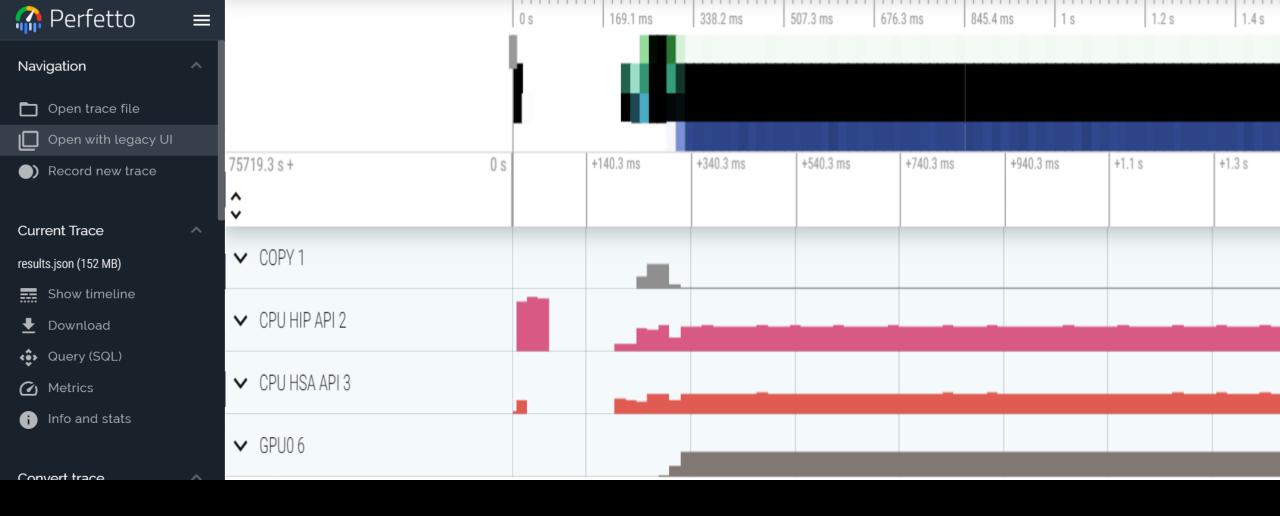
#### ATPESC 2022 – Tools Track

Suyash Tandon, Justin Chang, Julio Maia, Noel Chalmers, Paul T. Bauman, Nicholas Curtis, Nicholas Malaya, Damon McDougall, Rene van Oostrum, Noah Wolfe



### Agenda

- 1. Profiling
- 2. Debugging







### **AMD GPU Profiling**



ROC-profiler (or simply rocprof) is the command line front-end for AMD GPU profiling library

 Repo: <u>https://github.com/ROCm-</u> <u>Developer-Tools/rocprofiler</u>



#### Provided in the ROCm releases

rocprof contains the central components allowing the collection of application tracing and counter collection

• Under constant development



The output of rocprof can be visualized using the chrome browser with chrome tracing or <u>Perfetto</u>

\*More information can be found in the official <u>ROCm documentation</u>

### **Getting started with rocprof**

#### To get help:

\$ /opt/rocm/bin/rocprof -h

#### Useful housekeeping flags:

- --timestamp <on|off>
- --basenames <on|off> : template parameters and argument types)
- -o <output csv file>
- -d <data directory>
- -t <temporary directory> : Change the directory
   /tmp are placed. This allows you to save these temporary files

: turn on/off truncating gpu kernel names (i.e., removing

: turn on/off gpu kernel timestamps

- : Direct counter information to a particular file name
- : Send profiling data to a particular directory
- : Change the directory where data files typically created in

#### Flags directing rocprofiler activity:

- -i <input.txt|.xml> : specify an input file (note the output files will be named input.\*)
- --hsa-trace
- --hip-trace
- --roctx-trace

- : to trace GPU Kernels, host HSA events and HIP memory copies.
- : to trace HIP API calls
- : to trace roctx markers

#### Advanced usage

-m <metric file> : Allows the user to define and collect custom metrics.
 See rocprofiler/test/tool/\*.xml on GitHub for examples

### Collecting information about the kernels with rocprof

/op	:/rocm,	/bin/	'rocprof	stats	basenames	on	<app< th=""><th>with</th><th>args&gt;</th><th></th></app<>	with	args>	
-----	---------	-------	----------	-------	-----------	----	------------------------------------------------------------	------	-------	--

• 2 . csv files generated

- results.csv : information per each call of kernel
- results.stats.csv
- : statistics grouped by each kernel

			Contents	of results.stats.csv
"Name",	"Calls",	"TotalDurationNs",	, "AverageNs",	"Percentage"
"LocalLaplacianKernel",	1000,	817737586 <b>,</b>	817737,	40.908259879301134
"JacobiIterationKernel",	1000,	699515425 <b>,</b>	699515 <b>,</b>	34.994060790890174
"NormKernel1",	1001,	454737348,	454283,	22.748756969583884
"HaloLaplacianKernel",	1000,	14561933,	14561,	0.7284773865206329
"NormKernel2",	1001,	12395374,	12382,	0.620092789636225
"amd_rocclr_fillBufferAligned.kd",	1,	7040,	7040,	0.0003521840679465600

Identify "hottest" kernels that consume the most time.

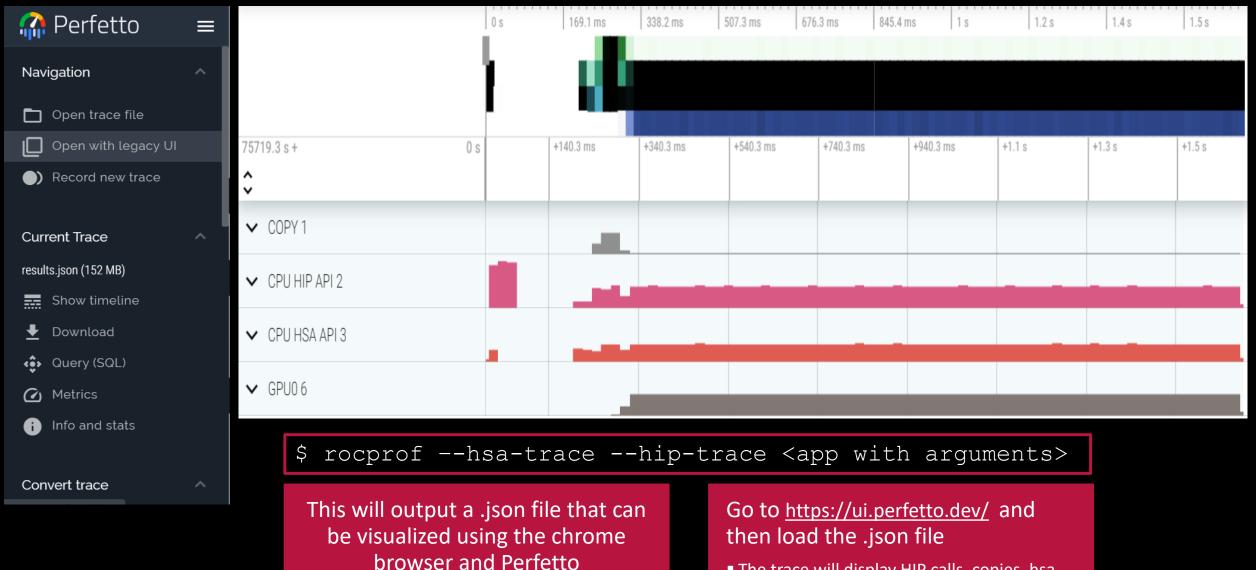
## Collecting traces with rocprof

rocprof can collect a variety of trace event types and generate timelines in JSON format for use with chrome-tracing or <u>Perfetto</u>

Multiple modes like --hip-trace and --has-trace can be combined

Trace Event	rocprof Trace Mode
HIP API call	hip-trace
GPU Kernels	hip-trace
Host <-> Device Memory copies	hip-trace
CPU HSA Calls	hsa-trace
User code markers	roctx-trace

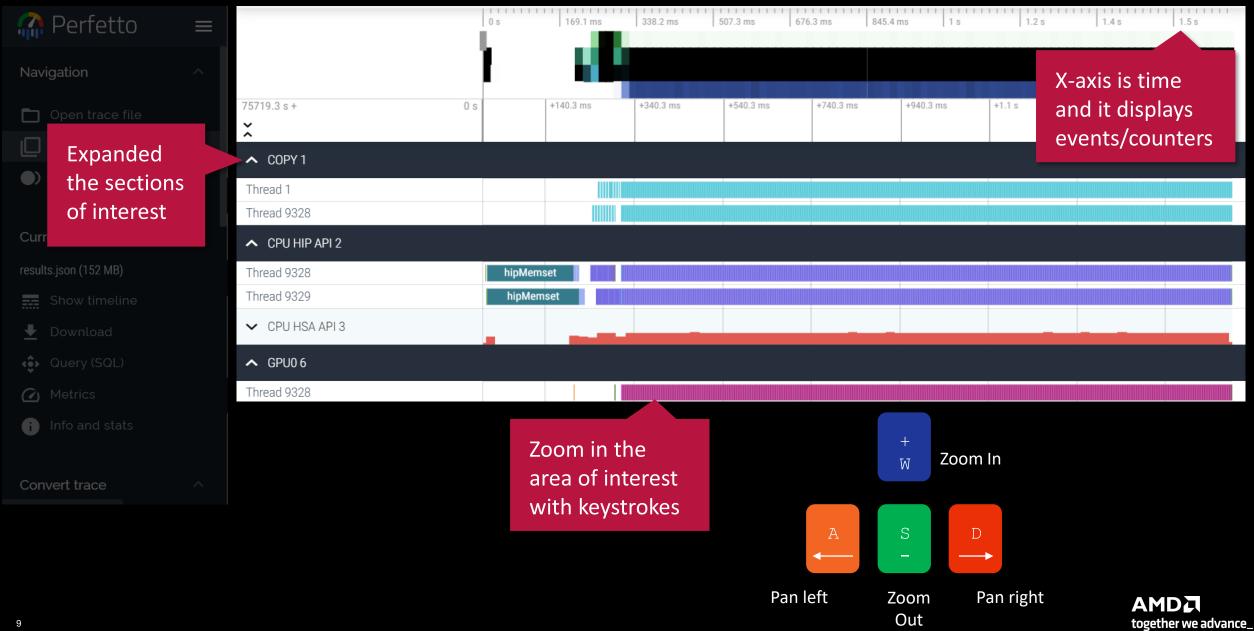




 The trace will display HIP calls, copies, hsa signals, and kernel calls

together we advance\_

8



🎧 Perfetto			Q Sea										<b>P</b>
Navigation		0 s	169.1 ms	338.2 ms	507.3 ms		76.3 ms 845.		l l l l l l l l l l l l l l l l l l l	1.2 s	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1.5 s	
Dpen trace file	75719.3 s + 312.6 ms	s +2.8 us	+102.8 us	+202.8 us	+302.8 us	+402.8 us	+502.8 us	+602.8 us	+702.8 us	+802.8 us	+902.8 us	+1 ms	+1.1 (
Open with legacy UI	▲ CPU HIP API 2			1				1					
Record new trace	Thread 9328	hipLa	aunchKernel				hipMemc		lemcpy				
	Thread 9329 CPU HSA API 3						прменс	ΥΥ Υ					ein i
Current Trace	Thread 9328	h	sa_execut <b>hsa</b>		hsa_signal_v	va			hsa_signal_wait_	scacquire			
results.json (152 MB)		h	hsa_signa	Lwai			hs	a_signal_wait_scac	quire				
Show timeline	Thread 9329												
Download	Thread 9344												
<ul> <li>Query (SQL)</li> <li>Metrics</li> </ul>	Thread 9346												
i Info and stats	▲ GPU0 6												
	Thread 9328				JacobilterationKern	el(int, double, d	double, double const*, dou	be Ope	en the ii	nformat	tion	*. dou	
Convert trace	Current Selection Flow Events							ban	el (bott	om righ	nt) to se	e 🗐	1^
Convert trace		Zoo	m and s	elect th	ne kerne	1				he kern			
			ee the li : enable										

together we advance\_

🔹 Query (SQL)		hsa_execut_ hsa hsa_signal_wa_		
C Metrics				
Info and stats				
onvert trace	Thread 9344			
	Thread 9328	JacobilterationKernel(int, double, double, double	ole const*, double const*, double*, dou NormKernel1 (int, double,	double, double const*, doub
	Current Selection Flow Events			† <b>Y</b>
	Slice Details			A
	Name	JacobilterationKernel(int, double, double, double const*, double const*, double*, double*) [clone .kd]		
	Category	null		
	Start time	312ms 848us 100ns		
	Duration	548us		
	Thread duration	0s (0.00%)		
	Thread	9328 GPU0 6		
	Process Slice ID	20238		
	args	20200		
	BeginNs	75719572538089		
	DurationNs	548641		
	EndNs	75719573086730		
	pid	9328		•



### **Collecting application traces with user-code markers**

/opt/rocm/bin/rocprof --hip-trace --roctx-trace <app with args>

MatrixTranspose.cpp example on roctracer GitHub page for sample usage

Code snippet					1								
		0 s	149.8	ms	299.7 ms	449.5 ms	59	).4 ms	749.2 ms	899.1 ms		1 s	1.2 s
<pre>roctracer_mark("before HIP LaunchKernel");</pre>													
••••													
roctxMark "before hipLaunchKernel");		.1 ms	+377.4 ns	+877.4 ns	+1.4 us	+1.9 us	+2.4 us	+2.9 us	+3.4 us	+3.9 us	+4.4 us	+4.9 us	+5.4 us
int rangeld =	· · · · · · · · · · · · · · · · · · ·												
<pre>roctxRangeStart("hipLaunchKernel range");</pre>	<ul> <li>Markers and Ranges 0</li> </ul>	********											
••••	Thread 0		*******	******						hipLa	unchKernel ran	je	
roctxRangePush("hipLaunchKernel");		••••	•••••	•••••	- before hi	ol aunchKernel				hi	pLaunchKernel		
hipLaunchKernelGGL(matrixtranspose,);	Thread 4092												
<pre>roctracer_mark("after HIP LaunchKernel");</pre>	▲ CPU HIP API 2												
<pre>roctxMark("after hipLaunchKernel");</pre>			M/	A D//	_				Li-Du-LO	allConfigurati			his Days 0
	Thread 4092		M#	ARK					nipPushC	allConfigurati			hipPopC
	▲ COPY 1												
	Thread 0												
	▲ GPU0 6												
	Thread 1												



### **Collecting hardware counters with rocprof**

Collect a number of hardware counters and derived counters

- \$/opt/rocm/bin/rocprof --list-basic
- \$/opt/rocm/bin/rocprof --list-derived

#### Specify counters in a counter file. For example:

- \$/opt/rocm/bin/rocprof -i my\_counters.txt <app with args>
- \$cat my counters.txt

pmc: Wavefronts VALUInsts VFetchInsts VWriteInsts VALUUtilization VALUBusy WriteSize
pmc: SALUInsts SFetchInsts LDSInsts FlatLDSInsts GDSInsts SALUBusy FetchSize
pmc: L2CacheHit MemUnitBusy MemUnitStalled WriteUnitStalled ALUStalledByLDS LDSBankConflict
...

- A limited number of counters can be collected during a specific pass of code
- Each line in the counter file will be collected in one pass
- You will receive an error suggesting alternative counter ordering if you have too many / conflicting counters on one line
- A . csv file will be created by this command containing all the requested counters

### **Commonly used counters**

VALUUtilization	<ul> <li>The percentage of ALUs active in a wave. Low VALUUtilization is likely due to high divergence or a poorly sized grid</li> </ul>
VALUBusy	<ul> <li>The percentage of GPUTime the vector ALU instructions are processed. Can be thought of as something like compute utilization</li> </ul>
FetchSize	<ul> <li>The total kilobytes fetched from global memory</li> </ul>
WriteSize	<ul> <li>The total kilobytes written to global memory</li> </ul>
L2CacheHit	<ul> <li>The percentage of fetch, write, atomic, and other instructions that hit the data in L2 cache</li> </ul>
MemUnitBusy	<ul> <li>The percentage of GPUTime the memory unit is active. The result includes the stall time</li> </ul>
MemUnitStalled	<ul> <li>The percentage of GPUTime the memory unit is stalled</li> </ul>
WriteUnitStalled	<ul> <li>The percentage of GPUTime the write unit is stalled</li> </ul>
Full list at	<ul> <li><u>https://github.com/ROCm-Developer-Tools/rocprofiler/blob/amd-master/test/tool/metrics.xml</u></li> </ul>

### Performance counters tips and tricks

#### GPU Hardware counters are global

- Kernel dispatches are serialized to ensure that only one dispatch is ever in flight
- It is recommended that no other applications are running that use the GPU when collecting performance counters

Use "--basenames on" which will report only kernel names, leaving off kernel arguments.

#### How do you time a kernel's duration?

- \$rocprof --timestamps on -i my\_counters.txt <app with args>
- This produces four times: DispatchNs, BeginNs, EndNs, and CompleteNs
- Closest thing to a kernel duration: EndNs BeginNs
- If you run with --stats the resultant results file will automatically include a column that calculates kernel duration
  - Note: the duration is aggregated over repeated calls to the same kernel

### **Collecting counters and traces**

#### on multiple MPI ranks

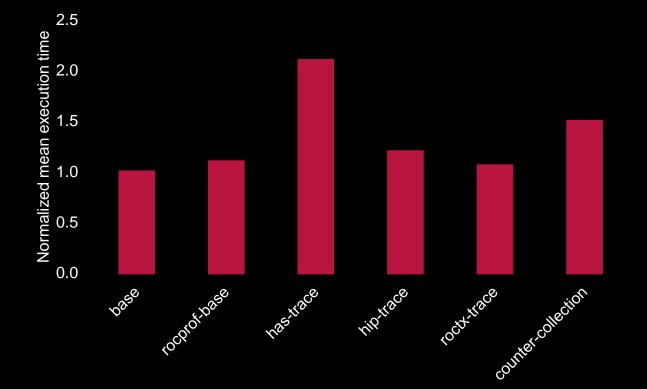
- rocprof can collect counters and traces for multiple MPI ranks.
- Say you want to profile an application usually called like this: mpiexec -np <n> <app with args>
- Then invoke the profiler by executing:

mpiexec -np <n> rocprof --hip-trace <app with args>

- This will produce a single unified CSV file for all ranks
- Multi-node profiling currently isn't supported

### **Profiling overhead**

Simple estimation of profiling overhead, obtained via wallclock timing of entire application run via Linux 'time' utility:



### Debugger

	Supp regular Konsole
File Ec	lit View Bookmarks Settings Help
	<pre>clude <hip hip_runtime.h=""></hip></pre>
2	
3C	pnstant float a = 1.0f;
4	
	lobal
	] saxpy(int n, float const* x, int incx, float* y, int incy)
7 {	
8	<pre>int i = blockDim.x*blockIdx.x + threadIdx.x;</pre>
10	if (i < n) y[i] += a*x[i];
11 }	y[1] += d*x[1],
12	
	main()
14 {	
15	int $n = 256$ ;
16	<pre>std::size_t size = sizeof(float)*n;</pre>
17	
18	float* d_x;
19	float* d_y;
20	hipMalloc(&d_x, size);
21	hipMalloc(&d_y, size);
22	
23	int num_groups = 2;
24	<pre>int group_size = 128;</pre>
25	<pre>saxpy&lt;&lt;<num_groups, group_size="">&gt;&gt;(n, d_x, 1, d_y, 1);</num_groups,></pre>
26	hipDeviceSynchronize();
/mnt/sha	ared/codes/saxpy/saxpy.hip.cpp

saxov · codb — Konsole

🔽 🖌

For help, type "help". Type "apropos word" to search for commands related to "word"... Reading symbols from [32m./saxpy[m... [?2004h(gdb)]

### Intro to ROCgdb

What is ROCgdb, from the tin:

The ROCm Debugger (ROCgdb) is the ROCm source-level debugger for Linux, based on the GNU Debugger (GDB). It enables heterogenous debugging on the ROCm platform of an x86-based host architecture along with AMD GPU architectures supported by the AMD Debugger API Library (ROCdbgapi). The AMD Debugger API Library (ROCdbgapi) is included with the ROCm release.

The current ROCm Debugger (ROCgdb) is an initial prototype that focuses on source line debugging and does not provide symbolic variable debugging capabilities. The user guide presents features and commands that may be implemented in future versions.

### Preparing the code for the debugger

Use any optimization level you like,

For example: -03

Have ROCm load code objects at initialization:

export HIP\_ENABLE\_DEFERRED\_LOADING=0

Add the flags:

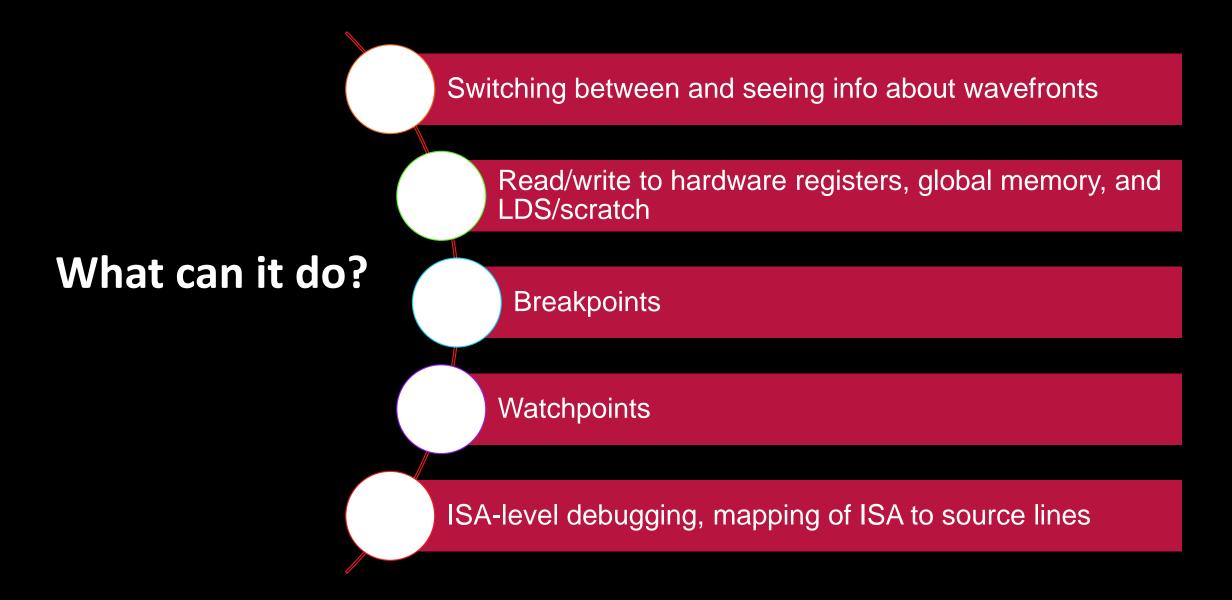
-ggdb

Optionally print even more useful information on API calls

export AMD LOG LEVEL=3

#### Example of what the compile options may look like...

mpic++ -I/usr/lib/x86\_64-linux-gnu/openmpi/include/openmpi –L/usr/lib/x86\_64-linuxgnu/openmpi/include -pthread –O3 –g **-ggdb** –fPIC –std=c++11 \ –march=native –Wall – I/opt/rocm/roctracer/include –I"/opt/rocm-4.2.0/hip/include" -I"/opt/rocm/llvm/bin/../lib/clang/12.0.0" –I/opt/rocm/hsa/include – I/opt/rocm/roctracer/include –c JacobiSetup.cpp -o JacobiSetup.o



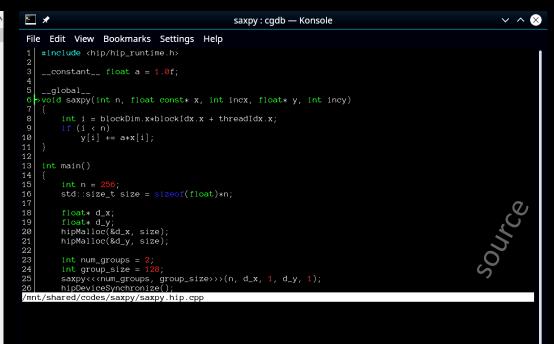


### Using a "GUI" with rocgdb

rocgdb -tui <app>

-saxpy.	:pp—		
	1 #i	nclude "hip/hip_runtime.h"	
	2 #i	nclude <stdio.h></stdio.h>	
	3		
	4 _	constant float a = 1.0f;	
	5		
}	6 _	global	
3	7 vc	id saxpy(int n, float const* x, int incx, float* y, int incy)	
	8 {		
	9	<pre>int i = blockIdx.x*blockDim.x + threadIdx.x;</pre>	
>	10	if $(i < n) y[i] = a*x[i] + y[i];$	
	11 }		
	12		
int main	13		
	14 {		
	15	int n = 256;	
	16	<pre>std::size = sizeof(float)*n;</pre>	
	17		
	18	float *d_x, *d_y;	-
	19	//hipMalloc(&d x, size);	<u>v</u>
	20	//hipMalloc(&d y, size);	,G
	21		
	22	int num groups= 2;	source
	23	int group size=128;	.0
	24	saxpy<< <num_groups,group_size>&gt;&gt;(n, d_x, 1, d_y, 1);</num_groups,group_size>	S
		00040) (((((((((((((((((((((((((((((((((	
		GPU Wave 1:2:1:1 In: saxpy	L10 PC: 0x7ffff7ec1094
		word" to search for commands related to "word"	
		s from saxpy	
(gdb) ru			
		am: /home/gmarkoma/saxpy	
		ing using libthread_db enabled]	
		thread_db library "/lib64/libthread_db.so.1".	
-		/7fffed428700 (LWP 11074)]	
Warning:	preci	se memory violation signal reporting is not enabled, reported	
location	may r	ot be accurate. See "show amdgpu precise-memory".	onsole
			S
Thread 3	"saxp	y" received signal SIGSEGV, Segmentation fault.	5
[Switchi	ng to	thread 3, lane 0 (AMDGPU Lane 1:2:1:1/0 (0,0,0)[0,0,0])]	.0

#### cgdb -d rocgdb <app>



[35;1mGNU gdb (rocm-rel-4.5-56) 11.1[m Copyright (C) 2021 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html> This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Type "show copying" and "show warranty" for details. This GDB was configured as "x86\_64-pc-linux-gnu". Type "show configuration" for configuration details. For bug reporting instructions, please see: <https://github.com/ROCm-Developer-Tools/ROCgdb/issues>. Find the GDB manual and other documentation resources online at: <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help". Type "apropos word" to search for commands related to "word"... Reading symbols from [32m./saxpy[m... [?2004h(gdb)

 $\mathbf{O}$ 

console

(gdb)

0x00007ffff7ec1094 in saxpy () at saxpy.cpp:10

### Setting a breakpoint in host code

Here we setup a breakpoint in the host code. We can inspect the device pointer and its values:

	jychang48@jychang48-workstatic	on: ~/Downloads/hiptutorial/hip	Q = -	- 🛛 😣
23 24 25 26 27 28	<pre>const int id_l = id - 1; const int id_r = id + 1; const int id_d = id - stride;</pre>	For help, type "help". Type "apropos word" to search for commands related to "word" Reading symbols from ./Jacobi_hip. (gdb) b Laplacian.cpp: 40		
28 29 30 31 32 33	(-U[id_d] + 2*U[id] - U[id_u])/(dy*dy); } }	Breakpoint 1 at 0x415080: file Laplacian.cpp, line 40.(gdb) run Starting program: /home/jychang48/Downloads/hiptutorial/nip/lacobi_hip [Thread debugging using libthread_db enabled] Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.sc [Detaching after fork from child process 129946]		
34 35 36 37	<pre>void LocalLaplacian(grid_t&amp; grid, mesh_t&amp; mesh,</pre>	[New Thread 0x7f2900108700 (LWP 129953)] [New Thread 0x7f28ff74a700 (LWP 129954)] Topology size: 1 x 1 Local domain size (current node): 4096 x 4096 Global domain size (all nodes): 4096 x 4096		
39 40 41 42	<pre>//there are (Nx-2)x(Ny-2) node on the interior of the mesh -&gt; int localNx = mesh.Nx-2; int localNy = mesh.Ny-2;</pre>	[New Thread 0x7f28f7f53700 (LWP 129955)] Rank 0 selecting device 0 on host jychang48-workstation [New Thread 0x7f28f71ff700 (LWP 129956)] [Thread 0x7f28f71ff700 (LWP 129956) exited]		
43 44 45 46	<pre>int ythreads = 16; dim3 threads(xthreads,ythreads,1);</pre>	[New Thread 0x7f28fc07f700 (LWP 129957)] [New Thread 0x7f28f7752700 (LWP 129958)] [New Thread 0x7f28f757f700 (LWP 129959)] Starting Jacobi run.		
47 48 49 50	<pre>(localNy+ythreads-1)/ythreads, 1); hipLaunchKernelGGL(LocalLaplacianKernel,</pre>	Iteration: 0 - Residual: 0.022108 Thread 1 Jacobi_hip hit_Breakpoint 1, LocalLaplacian (grid=, mesk x69a6e0, d_U=0x7f27a7e00000, d_4U=0x7f279fc000000) at Laplacian.cpp:40		tream=0
51 52 53 54	threads, 0, stream, localNx, localNy, mesh.Nx,	(gdb) p d U \$1 = (double *) 0x7f27a7e00000 (gdb) p d U[0] \$2 = 0		
55 56 <mark>/home</mark>	d_U, d_AU);	(gdb) p d U[0]@10 \$3 = {0, 0, 0, 0, 0, 0, 0, 0, 0} (gdb)		

### Setting a breakpoint in device kernel

#### Invoke 'b' or 'break' to the device kernel of interest:

	jychang48@jychang48-workstat	ion: ~/Downloads/hiptutorial/hip	Q = -	
1 2 3 4 5 6 7 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	<pre>//***********************************</pre>	Topology size: 1 x 1 Local domain size (current node): 4096 x 4096 Global domain size (all nodes): 4096 x 4096 [New Thread 0x7f28f7f53700 (LWP 129955)] Rank 0 selecting device 0 on host jychang48-workstation [New Thread 0x7f28f7lff700 (LWP 129956)] [Thread 0x7f28f7lff700 (LWP 129956) exited] [New Thread 0x7f28f707f700 (LWP 129957)] [New Thread 0x7f28f757f700 (LWP 129958)] [New Thread 0x7f28f757f700 (LWP 129959)] Starting Jacobi run. Iteration: 0 - Residual: 0.022108 Thread 1 "Jacobi hip" hit Breakpoint 1, LocalLaplacian (grid=, mesh= x69a6e0, d U=0x7f27a7e00000, d_AU=0x7f279fc00000) at Laplacian.cpp:40 (gdb) p d U \$1 = (double *) 0x7f27a7e00000 (gdb) p d U[0] \$2 = 0 (gdb) p d U[0]@10 \$3 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0} (gdb) step (gdb) step (gdb) b LocalLaplacianKernel Function LocalLaplacianKernel" not defined.		
23 24	<pre>const int id l = id - 1;</pre>	(gdb) step (gdb) b LocalLaplacianKernel		
26 27	<pre>const int id_d = id - stride; const int id u = id + stride;</pre>	Make breakpoint pending on future shared library load? (y or [n]) y		
28 29 30	AU[id] = (-U[id l] + 2*U[id] - U[id r])/(dx*dx) +	Breakpoint 2 (LocalLaplacianKernel) pending.(gdo) continue Continuing. [Switching to AMDGPU Thread 1:5:1:1 (0,0,0)/0]		
30 31 32 33 34		Thread 9 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=< >, localNy= <optimized out="">, stride=<optimized out="">, dx=<optimized out="">, d out&gt;, U=<optimized out="">, AU=<optimized out="">) at Laplacian.cpp:9</optimized></optimized></optimized></optimized></optimized>		
/hom	me/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp	(gdb)		

### Examine switching context to new thread

What happens when you type 'step'? Another thread hit the same breakpoint! GDB will switch context to the new thread:

	jychang48@jychang48-worksta	tion: ~/Downloads/hiptutorial/hip	Q ≡ -	
1 2 2	<pre>//***********************************</pre>	\$2 = 0  (gdb) p d_U[0]@10		
3 4 5	<pre>#include "Jacobi.hpp"</pre>	\$3 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0} (gdb) step (gdb) step		
6	#Include Successingp	(gdb) b LocalLaplacianKernel		
7	// AU i,j = (-U i+1,j + 2U i,j - U i-1,j)/dx^2 +	Function "LocalLaplacianKernel" not defined.		
8	//(-U_i,j+1 + 2U_i,j - U_i,j-1)/dy^2	Make breakpoint pending on future shared library load? (y or [n]) y		
9	<pre>&gt;global void LocalLaplacianKernel(const int localNx,</pre>			
10	const int localNy,	Breakpoint 2 (LocalLaplacianKernel) pending.(gdb) continue		
11	const int stride,	Continuing.		
12	const dfloat dx,	[Switching to AMDGPU Thread 1:5:1:1 (0,0,0)/0]		
13 14	const dfloat dy,	  Thread 9 "Jacobi hip" hit Breakpoint 2, LocalLaplacianKernel (localNx	- contimizo	d out
15	<pre>const dfloat *_restrict U,</pre>	<pre>&gt;, localNy=<optimized out="">, stride=<optimized out="">, dx=<optimized out<="" pre=""></optimized></optimized></optimized></pre>		
16		d out>, U= <optimized out="">, AU=<optimized out="">) at Laplacian.cpp:9</optimized></optimized>	, dy-topt	111120
17	<pre>const int i = threadIdx.x+blockIdx.x*blockDim.x;</pre>	(qdb) step		
18	<pre>const int j = threadIdx.y+blockIdx.y*blockDim.y;</pre>	[Switching to AMDGPU Thread 1:5:1:1287 (65,1,0)/2]		
19				
20	<pre>if ((i<localnx) &&="" (j<localny))="" pre="" {<=""></localnx)></pre>	Thread 1295 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (loca		
21		<pre>out&gt;, localNy=<optimized out="">, stride=<optimized out="">, dx=<optimized< pre=""></optimized<></optimized></optimized></pre>	out>, dy=<	optim
22	<pre>const int id = (i+1) + (j+1)*stride;</pre>	<pre>ized out&gt;, U=<optimized out="">, AU=<optimized out="">) at Laplacian.cpp:9</optimized></optimized></pre>		
23 24	const int id $l = id - 1$ ;	(gdb) step [Switching to AMDGPU Thread 1:5:1:1269 (61,1,0)/0]		
25	const int id $r = id + 1;$			
26	const int id d = id - stride;	Thread 1277 "Jacobi hip" hit Breakpoint 2, LocalLaplacianKernel (loca	lNx= <optim< td=""><td>ized</td></optim<>	ized
27	<pre>const int id u = id + stride;</pre>	out>, localNy= <optimized out="">, stride=<optimized out="">, dx=<optimized< td=""><td></td><td></td></optimized<></optimized></optimized>		
28		ized out>, U= <optimized out="">, AU=<optimized out="">) at Laplacian.cpp:9</optimized></optimized>		·
29	AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +	(gdb) step		
30	(-U[id_d] + 2*U[id] - U[id_u])/(dy*dy);	[Switching to AMDGPU Thread 1:5:1:363 (90,0,0)/2]		
31 32				
32 33	}	Thread 371 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (local		
	<pre>void LocalLaplacian(grid_t&amp; grid, mesh_t&amp; mesh,</pre>	<pre>ut&gt;, localNy=<optimized out="">, stride=<optimized out="">, dx=<optimized o<br="">zed out&gt;, U=<optimized out="">, AU=<optimized out="">) at Laplacian.cpp:9</optimized></optimized></optimized></optimized></optimized></pre>	ut>, ay= <o< td=""><td>primi</td></o<>	primi
/hom	e/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp	(gdb)		
TOIL	c/ j y chang+o, bown coads/ http://control/http//cap/actum.cpp			



### Examine switching context to new thread

AMDGPU Thread agent-id:queue-id:dispatch-num:wave-id (work-group-z,work-group-y,work-group-x)/work-group-thread-index

#### [Switching to AMDGPU Thread 1:5:1:1 (0,0,0)/0]

Thread 9 "Jacobi\_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized out >, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optimize d out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9

A Province	<u> </u>			
agent-id	queue-id	dispatch-num	wave-id	(z, y, x)
Agent Target ID	Queue Target ID	Dispatch Target ID – how many kernels have been launched	Wavefront ID – index of wavefront of kernel	work-group/block index

### Examine the ISA with rocgdb using cgdb

#### Using cgdb, type ESC -> :set dis -> ENTER

	jychang48@jychang48-workstati	on: ~/Downloads/hiptutorial/hip	Q = - • ×
3 4		Make breakpoint pending on future shared library load? (y or [n]) y	
	0x00007f28f722f000 <+0>: s_load_dwordx4 s[0:3], s[6:7], 0x0	Deceloriet 2 (Leceller) - significantly mending (adh) - setimus	
5		Breakpoint 2 (LocalLaplacianKernel) pending.(gdb) continue	
6 7	<pre>/opt/rocm-4.2.0/hip/include/hip/amd_detail/hip_runtime.h: 232 return ockl get local size(x);</pre>	Continuing. [Switching to AMDGPU Thread 1:5:1:1 (0,0,0)/0]	
8	$0 \times 0.000757255008 < 19 \times 10^{-1}$ s load dword s10 s [4:5] $0 \times 4$		
8 9	0x00007f28f722f010 <+16>: s waitcnt lgkmcnt(0)	Thread 9 "Jacobi hip" hit Breakpoint 2, LocalLaplacianKernel (localNx	<pre>&lt;<pre>c=<optimized out<="" pre=""></optimized></pre></pre>
10	0x00007f28f722f014 <+20>: s lshr b32 s3, s10, 16	<pre>&gt;, localNy=<optimized out="">, stride=<optimized out="">, dx=<optimized out<="" pre=""></optimized></optimized></optimized></pre>	
11		d out>, U= <optimized out="">, AU=<optimized out="">) at Laplacian.cpp:9</optimized></optimized>	
12	0x00007f28f722f020 <+32>: s_mul_i32 s8, s8, s4	(gdb) step	
13	0x00007f28f722f024 <+36>: s_mul_i32 s9, s9, s3	[Switching to AMDGPU Thread 1:5:1:1287 (65,1,0)/2]	
14			
15 16		Thread 1295 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (loca	
10		<pre>out&gt;, localNy=<optimized out="">, stride=<optimized out="">, dx=<optimized ized out&gt;, U=<optimized out="">, AU=<optimized out="">) at Laplacian.cpp:9</optimized></optimized></optimized </optimized></optimized></pre>	out>, dy= <optim< td=""></optim<>
18	0x0000712017221020 <+=0>. V_auu_us2_es2 v0, s0, v0	(gdb) step	
19	<pre>18 const int j = threadIdx.y+blockIdx.y*blockDim.y;</pre>	[Switching to AMDGPU Thread 1:5:1:1269 (61,1,0)/0]	
20-	—> 0x00007f28f722f02c <+44>: v add u32 e32 v1, s9, v1		
21		Thread 1277 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (loca	
22		out>, localNy= <optimized out="">, stride=<optimized out="">, dx=<optimized< td=""><td>out&gt;, dy=<optim< td=""></optim<></td></optimized<></optimized></optimized>	out>, dy= <optim< td=""></optim<>
23		ized out>, U= <optimized out="">, AU=<optimized out="">) at Laplacian.cpp:9</optimized></optimized>	
24	0x00007f28f722f030 <+48>: v_cmp_gt_i32_e32 vcc, s0, v0	(gdb) step	
25 26		[Switching to AMDGPU Thread 1:5:1:363 (90,0,0)/2]	
20	0x00007f28f722f03c <+60>: s_and_b64 s[0:1], vcc, s[0:1] 0x00007f28f722f040 <+64>: s_and_saveexec_b64 s[4:5], s[0:1]	Thread 371 "Jacobi hip" hit Breakpoint 2, LocalLaplacianKernel (local	Ny-contimized o
28	0x00007f28f722f044 <+68>: s cbranch execz 104	ut>, localNy= <optimized out="">, stride=<optimized out="">, dx=<optimized o<="" th=""><th></th></optimized></optimized></optimized>	
29		zed out>, U= <optimized out="">, AU=<optimized out="">) at Laplacian.cpp:9</optimized></optimized>	
30	21	(gdb) disable 2	
30 31	<pre>22</pre>	(gdb) step	
32 33	0x00007f28f722f048 <+72>: v_add_u32_e32 v1, 1, v1	(gdb) stepi	
33		<pre>0x00007f28f722f010 inHIP_BlockDim::operator() (this=<optimized out<="" pre=""></optimized></pre>	
34	0x00007f28f722f054 <+84>: s_load_dwordx2 s[0:1], s[6:7], 0x20	out>) at /opt/rocm-4.2.0/hip/include/hip/amd_detail/hip_runtime.h:23	32
35	•	(gdb) step	
36	<b>9</b> global void LocalLaplacianKernel(const int localNx,	(gdb) step	
>ae	for function _Z20LocalLaplacianKerneliiiddPKdPd: (7f28f722f000 - 7f28f722f1e8) **	(gab)	

### Switching wavefronts

#### Use info threads to see the location of both host threads and GPU wavefronts

	jychang48@jychang48-workstati	on: ~/Downloads/hiptutorial/hip	Q = - 0 😣
15	Laplacian.cpp:	(gdb) info threads	[110/112]
16	<pre>17 const int i = threadIdx.x+blockIdx.x*blockDim.x;</pre>	Id Target Id Frame	
17	0x00007f28f722f028 <+40>: v add u32 e32 v0, s8, v0	1 Thread 0x7f291814d8c0 (LWP 129937) "Jacobi hip" 0x000	07f2918b7a89b in sched y
18		ield () from /lib/x86 64-linux-gnu/libc.so.6	
19	<pre>18 const int j = threadIdx.y+blockIdx.y*blockDim.y;</pre>	2 Thread 0x7f2900108700 (LWP 129953) "Jacobi hip" 0x000	07f2918b8aaff in poll ()
20	0x00007f28f722f02c <+44>: v_add_u32_e32 v1, s9, v1	from /lib/x86 64-linux-gnu/libc.so.6	
21		3 Thread 0x7f28ff74a700 (LWP 129954) "Jacobi_hip" 0x000	07f2918b975ce in epoll w
22		ait () from /lib/x86 64-linux-gnu/libc.so.6	' –
23	20 if ((i <localnx) &&="" (i<localnv))="" td="" {<=""><td>4 Thread 0x7f28f7f53700 (LWP 129955) "Jacobi hip" 0x000</td><td>07f2918b8c50b in ioctl (</td></localnx)>	4 Thread 0x7f28f7f53700 (LWP 129955) "Jacobi hip" 0x000	07f2918b8c50b in ioctl (
24	0x00007f28f722f030 <+48>: v_cmp_gt_i32_e32 vcc, s0, v0	) from /lib/x86 64-linux-gnu/libc.so.6	
25	0x00007f28f722f034 <+52>: v cmp gt i32 e64 s[0:1], s1, v1	6 Thread 0x7f28fc07f700 (LWP 129957) "Jacobi hip" 0x000	07f2918b7a89b in sched y
26	<pre>0x00007f28f722f034 &lt;+52&gt;: v_cmp_gt_i32_e64 s[0:1], s1, v1 0x00007f28f722f03c &lt;+60&gt;: s_and_b64 s[0:1], vcc, s[0:1] 0x00007f28f722f040 &lt;+64&gt;: s_and_saveexec_b64 s[4:5], s[0:1] 0x00007f28f722f044 &lt;+68&gt;: s_cbranch_execz 104</pre>	ield () from /lib/x86 64-linux-gnu/libc.so.6	
27	0x00007f28f722f040 <+64>: s and saveexec b64 s[4:5], s[0:1]	7 Thread 0x7f28f7752700 (LWP 129958) "Jacobi hip" 0x000	07f2918b8c50b in ioctl (
28	0x00007f28f722f044 <+68>: s cbranch execz 104	) from /lib/x86 64-linux-gnu/libc.so.6	
29		8 Thread 0x7f28f757f700 (LWP 129959) "Jacobi hip" 0x000	07f291a2f4678 in do fute
30		x wait.constprop () from /lib/x86 64-linux-gnu/libpthread.so	.0
31	<pre>22</pre>	* 371 AMDGPU Thread 1:5:1:363 (90,0,0)/2 "Jacobi_hip" Local	LaplacianKernel (localNx
	-> 0x00007f28f722f048 <+72>: v_add_u32_e32 v1, 1, v1	<pre>=<optimized out="">, localNy=<optimized out="">, stride=<optimized< pre=""></optimized<></optimized></optimized></pre>	
33	0x000071281722104c <+/6>: v_mul_lo_u32 v1, v1, s2	>, dy= <optimized out="">, U=<optimized out="">, AU=<optimized out=""></optimized></optimized></optimized>	•) at Laplacian.cpp:22 🌙
34	0x00007f28f722f054 <+84>:	Cgdb)	
35		Id Target Id Frame	
36	9global void LocalLaplacianKernel(const int localNx,	1 Thread 0x7f291814d8c0 (LWP 129937) "Jacobi_hip" 0x000	07f2918b7a89b in sched_y
37	0x00007f28f722f05c <+92>: s_load_dwordx4 s[8:11], s[6:7], 0x10	ield () from /lib/x86_64-linux-gnu/libc.so.6	
38		2 Thread 0x7f2900108700 (LWP 129953) "Jacobi_hip" 0x000	07f2918b8aaff in poll ()
39		from /lib/x86_64-linux-gnu/libc.so.6	
40	<pre>22</pre>	3 Thread 0x7f28ff74a700 (LWP 129954) "Jacobi_hip" 0x000	07f2918b975ce in epoll_w
41	0x00007f28f722f064 <+100>: v_add_u32_e32 v4, v1, v0	ait () from /lib/x86_64-linux-gnu/libc.so.6	
42		4 Thread 0x7f28f7f53700 (LWP 129955) "Jacobi_hip" 0x000	07f2918b8c50b in ioctl (
43		) from /lib/x86_64-linux-gnu/libc.so.6	
44	29 $AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +$	6 Thread 0x7f28fc07f700 (LWP 129957) "Jacobi_hip" 0x000	07f2918b7a89b in sched_y
45	<b>0x00007f28f722f068 &lt;+104&gt;:</b>	ield () from /lib/x86_64-linux-gnu/libc.so.6	
46	0x00007f28f722f06c <+108>: v_lshlrev_b64 v[5:6], 3, v[4:5]	7 Thread 0x7f28f7752700 (LWP 129958) "Jacobi_hip" 0x000	07f2918b8c50b in ioctl (
47	0x00007f28f722f074 <+116>: s_waitcnt lgkmcnt(0)	) from /lib/x86_64-linux-gnu/libc.so.6	
48	<b>0x00007f28f722f078 &lt;+120&gt;:</b> v mov b32 e32 v15, s1	8 Thread 0x7f28f757f700 (LWP 129959) "Jacobi_hip" 0x000	
>de f	or function _Z20LocalLaplacianKerneliiiddPKdPd: (7f28f722f000 - 7f28f722f1e8) **	x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so	.0

### Switching wavefronts

#### Or use thread <tid> to examine one particular thread

	jychang48@jychang48-workstati	on: ~/Downloads/hiptutorial/hip	Q = - • 😣
15	Laplacian.cpp:	(gdb) info threads	[110/112]
16	<pre>17 const int i = threadIdx.x+blockIdx.x*blockDim.x;</pre>	Id Target Id F	rame
17	0x00007f28f722f028 <+40>: v add u32 e32 v0, s8, v0	1 Thread 0x7f291814d8c0 (LWP 129937) "Jacobi hip" 0	x00007f2918b7a89b in sched y
18		ield () from /lib/x86 64-linux-gnu/libc.so.6	
19	<pre>18 const int j = threadIdx.y+blockIdx.y*blockDim.y;</pre>	2 Thread 0x7f2900108700 (LWP 129953) "Jacobi hip" 0	x00007f2918b8aaff in poll ()
20	0x00007f28f722f02c <+44>: v add u32 e32 v1, s9, v1	from /lib/x86 64-linux-gnu/libc.so.6	
21		3 Thread 0x7f28ff74a700 (LWP 129954) "Jacobi hip" 0	x00007f2918b975ce in epoll w
22		ait () from /lib/x86 64-linux-gnu/libc.so.6	· -
23	20 if ((i <localnx) &&="" (j<localny))="" td="" {<=""><td>4 Thread 0x7f28f7f53700 (LWP 129955) "Jacobi hip" 0</td><td>x00007f2918b8c50b in ioctl (</td></localnx)>	4 Thread 0x7f28f7f53700 (LWP 129955) "Jacobi hip" 0	x00007f2918b8c50b in ioctl (
24	0x00007f28f722f030 <+48>: v cmp gt i32 e32 vcc, s0, v0	) from /lib/x86 64-linux-gnu/libc.so.6	
25	0x00007f28f722f034 <+52>: v cmp gt i32 e64 s[0:1], s1, v1	6 Thread 0x7f28fc07f700 (LWP 129957) "Jacobi hip" 0	x00007f2918b7a89b in sched y
26		ield () from /lib/x86 64-linux-gnu/libc.so.6	
27	0x00007f28f722f040 <+64>: s_and_saveexec_b64 s[4:5], s[0:1]	7 Thread 0x7f28f7752700 (LWP 129958) "Jacobi_hip" 0	x00007f2918b8c50b in ioctl (
28	0x00007f28f722f044 <+60>: s cbranch_execz 104	) from /lib/x86_64-linux-gnu/libc.so.6	
29		8 Thread 0x7f28f757f700 (LWP 129959) "Jacobi_hip" 0	x00007f291a2f4678 in do_fute
30		x_wait.constprop () from /lib/x86_64-linux-gnu/libpthrea	
31	<pre>22</pre>	* 371 AMDGPU Thread 1:5:1:363 (90,0,0)/2 "Jacobi_hip" L	
32 -		= <optimized out="">, localNy=<optimized out="">, stride=<optim< td=""><td></td></optim<></optimized></optimized>	
33		<pre>&gt;, dy=<optimized out="">, U=<optimized out="">, AU=<optimized< pre=""></optimized<></optimized></optimized></pre>	out>) at Laplacian.cpp:22
34	0x00007f28f722f054 <+84>: s_load_dwordx2 s[0:1], s[6:7], 0x20	(Եցվի)	
35 36			rame
36	9global void LocalLaplacianKernel(const int localNx,	1 Thread 0x7f291814d8c0 (LWP 129937) "Jacobi_hip" 0	x00007f2918b7a89b in sched_y
37	0x00007f28f722f05c <+92>: s_load_dwordx4 s[8:11], s[6:7], 0x10	ield () from /lib/x86_64-linux-gnu/libc.so.6	
38		2 Thread 0x7f2900108700 (LWP 129953) "Jacobi_hip" 0	x00007f2918b8aaff in poll ()
39 40		from /lib/x86_64-linux-gnu/libc.so.6	
40	<pre>22</pre>	3 Thread 0x7f28ff74a700 (LWP 129954) "Jacobi_hip" 0	x00007f2918b975ce in epoll_w
41	0x00007f28f722f064 <+100>: v_add_u32_e32 v4, v1, v0	ait () from /lib/x86_64-linux-gnu/libc.so.6	
42		4 Thread 0x7f28f7f53700 (LWP 129955) "Jacobi_hip" 0	x00007f2918b8c50b in ioctl (
43	28	) from /lib/x86_64-linux-gnu/libc.so.6	
44	29 $AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +$	6 Thread 0x7f28fc07f700 (LWP 129957) "Jacobi_hip" 0	x00007f2918b7a89b in sched_y
45	0x00007f28f722f068 <+104>: v_ashrrev_i32_e32 v5, 31, v4	ield () from /lib/x86_64-linux-gnu/libc.so.6	
46	0x00007f28f722f06c <+108>: v_lshlrev_b64 v[5:6], 3, v[4:5]	7 Thread 0x7f28f7752700 (LWP 129958) "Jacobi_hip" 0	x00007†2918b8c50b in ioctl (
47	<pre>0x00007f28f722f074 &lt;+116&gt;: s_waitcnt lgkmcnt(0)</pre>	) from /lib/x86_64-linux-gnu/libc.so.6	
48	0x00007f28f722f078 <+120>: v_mov_b32_e32_v15, s1	8 Thread 0x7f28f757f700 (LWP 129959) "Jacobi_hip" 0	x00007f291a2f4678 in do_fute
>de 1	for function _Z20LocalLaplacianKerneliiiddPKdPd: (7f28f722f000 - 7f28f722f1e8) **	x_wait.constprop () from /lib/x86_64-linux-gnu/libpthrea	ld.so.0

### Other tips and tricks with rocgdb

#### Use export AMD\_LOG\_LEVEL=3 to print all API calls and more happening

	jychang48@jychang48-workstati	on: ~/Downloads/hiptutorial/hip Q = -	- 🗆 😣
23			41/144]
24	<pre>const int id_l = id - 1;</pre>	(gdb) b Laplacian.cpp: 40	
25	<pre>const int id_r = id + 1;</pre>		
26	<pre>const int id d = id - stride; const int id u = id u stride;</pre>	Breakpoint 1 at 0x415080: file Laplacian.cpp, line 40.(gdb) run	
27 28	<pre>const int id_u = id + stride;</pre>	Starting program: /home/jychang48/Downloads/hiptutorial/hip/Jacobi_hip -g 1 1 [Thread debugging using libthread db enabled]	
28 29	AU[id] = (-U[id l] + <b>2</b> *U[id] - U[id r])/(dx*dx) +	Using host libthread db library "/lib/x86 64-linux-gnu/libthread db.so.1".	
30	(-U[id d] + 2*U[id] - U[id u])/(dy*dy);		457 : 4
31	}	3543864078 us: Initializing HSA stack.	57.4
32	}	:3:comgrctx.cpp :33 : 43543879647 us: Loading COMGR library.	
33	<b>,</b>	:3:comgrctx.cpp :33 : 43543879647 us: Loading COMGR library. :3:rocdevice.cpp :200 : 43543882744 us: Numa selects cpu agent[0]=0	9x55e1a
34	void LocalLaplacian(grid t& grid, mesh t& mesh,	d860740(fine=0x55e1ad8490e0,coarse=0x55e1ad7e32c0, kern arg=0x55e1ad8d7b20) fo	
35	hipStream t stream,	agent=0x7ffa27a13769	51
36	dfloat* d_U,		
37	dfloat* d_AU) {	[New Thread 0x7f39eab37700 (LWP 151591)]	
38		[New Thread 0x7f39ea179700 (LWP 151592)]	
39	<pre>//there are (Nx-2)x(Ny-2) node on the interior of the mesh</pre>	Topology size: 1 x 1	
		Local domain size (current node): 4096 x 4096	
41	<pre>int localNy = mesh.Ny-2;</pre>	Global domain size (all nodes): 4096 x 4096	
42	dut utburged a to	:3:rocdevice.cpp :457 : 43544254545 us: Initializing HSA stack.	
43	<pre>int xthreads = 16; int ythreads = 16;</pre>	[New Thread 0x7f39e253e700 (LWP 151593)]	
44 45	<pre>int ythreads = 16;</pre>	:3:comgrctx.cpp :33 : 43544267424 us: Loading COMGR library. :3:rocdevice.cpp :200 : 43544292744 us: Numa selects cpu agent[0]=0	aveesh4
46	<pre>dim3 threads(xthreads,ythreads,1);</pre>	0(fine=0x84b9c0,coarse=0x8a39a0, kern arg=0x8a2900) for gpu agent=0x7f3a03a431	140
47	dim3 blocks((localNx+xthreads-1)/xthreads,	:3:hip context.cpp :124 : 43544298171 us: 151575: [7f3a02b7c8c0] hipI	Init: R
48		eturned hipSuccess :	
49		:3:hip_device_runtime.cpp :497 : 43544298229 us: 151575: [7f3a02b7c8c0] hipG	GetDevi
50	hipLaunchKernelGGL(LocalLaplacianKernel,	ceCount ( 0x7ffd1902c1a4 )	
51	blocks,	:3:hip device runtime.cpp :499 : 43544298236 us: 151575: [7f3a02b7c8c0] hipG	GetDevi
50 51 52 53 54 55	threads,	ceCount: Returned hipSuccess :	
53	0, stream,	:3:hip_device_runtime.cpp	SetDevi
54	localNx, localNy, mesh.Nx,	ce ( 0 )	
55	mesh.dx, mesh.dy,	:3:hip_device_runtime.cpp	SetDevi
56	d_U, d_AU);	ce: Returned hipSuccess :	
/home	/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp	Rank 0 selecting device 0 on host jychang48-workstation	

### Other tips and tricks with rocgdb

#### Use export AMD\_LOG\_LEVEL=3 to print all API calls and more happening

In jychang48@jychang48-workstati	ion: ~/Downloads/hiptutorial/hip 🛛 🔍 📃 – 🗉 😣
23 24 const int id_l = id - 1; 25 const int id r = id + 1;	Rank 0 selecting device 0 on host jychang48-workstation         [61/144]           :3:hip_memory.cpp         :289 : 43544298319 us: 151575: [7f3a02b7c8c0] hipHostMal           loc ( 0x7ffd1902c6f8, 131072, 0 )
<pre>23 24 25 25 26 27 28 29 30 32 32 32 33 23 23 23 23 23 23 23 23 23</pre>	:3:hip_memory.cpp :318 : 43544298463 us: 151575: [7f3a02b7c8c0] hipHostMal loc: Returned hipSuccess : 0x7f39e85c0000: duration: 144 us :3:hip_memory.cpp :289 : 43544298473 us: 151575: [7f3a02b7c8c0] hipHostMal
29 AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) + 30 (-U[id_d] + 2*U[id] - U[id_u])/(dy*dy);	loc( 0x7ffd1902c700, 131072, 0 ) :3:hip memory.cpp     :318 : 43544298589 us: 151575: [7f3a02b7c8c0] hipHostMal
31 } 32 } 33	loc: Returned hipSuccess : 0x7f39e8580000: duration: 116 us :3:hip_memory.cpp :283 : 43544298607 us: 151575: [7f3a02b7c8c0] hipMalloc ( 0x7ffd1902c708, 131072 )
<pre>34 void LocalLaplacian(grid_t&amp; grid, mesh_t&amp; mesh, 35 hipStream_t stream, 36 dfloat* d_U, 37 dfloat* d_AU) { 38 39 //there are (Nx-2)x(Ny-2) node on the interior of the mesh</pre>	:3:rocdevice.cpp :2065: 43544298694 us: device=0x8d2890, freeMem_ = 0xfef e0000 :3:hip memory.cpp :285 : 43544298702 us: 151575: [7f3a02b7c8c0] hipMalloc:
37 dfloat* d_AU) { 38	Returned hipSuccess : 0x7f39e1a000000: duration: 95 us :3:hip memory.cpp :1877: 43544298712 us: 151575: [7f3a02b7c8c0] hipMemset
40 -> int localNx = mesh.Nx-2;	( 0x7f39ela00000, 0, 131072 ) :3:rocdevice.cpp :2566: 43544299081 us: number of allocated hardware queu es with low priority: 0, with normal priority: 0, with high priority: 0, maximum per
42 43 int xthreads = 16;	priority is: 4 [New Thread 0x7f39e19ff700 (LWP 151594)]
<pre>44 int ythreads = 16; 45 46 dim3 threads(xthreads,ythreads,1);</pre>	[Thread 0x7f39e19ff700 (LWP 151594) exited] [New Thread 0x7f39e857f700 (LWP 151595)] :3:rocdevice.cpp :2638: 43544316233 us: created hardware queue 0x7f39e88c
<pre>47 dim3 blocks((localNx+xthreads-1)/xthreads, 48 (localNy+ythreads-1)/ythreads, 1); 49 (localNy+ythreads-1)/ythreads, 1);</pre>	6000 with size 1024 with priority 1, cooperative: 0 :3:devprogram.cpp :2463: 43544489762 us: Using Code Object V4.
49 50 hipLaunchKernelGGL(LocalLaplacianKernel, 51 blocks,	:3:rocvirtual.cpp :572 : 43544495580 us: ! arg1: uint* bufUInt = ptr:0x 7f39e1a00000 obj:[0x7f39e1a00000-0x7f39e1a20000] threadId : 7f39e857f700
<pre>41 int localNy = mesh.Ny-2; 42 43 int xthreads = 16; 44 int ythreads = 16; 45 46 dim3 threads(xthreads,ythreads,1); 47 dim3 blocks((localNx+xthreads-1)/xthreads, 48 (localNy+ythreads-1)/ythreads, 1); 49 50 hipLaunchKernelGGL(LocalLaplacianKernel, 51 blocks, 52 threads, 53 0, stream, 54 localNx, localNy, mesh.Nx, 55 mesh.dx, mesh.dy, 56 d U, d AU);</pre>	:3:rocvirtual.cpp :572 : 43544495588 us: ! arg2: uchar* pattern = ptr:0 x7f39e876e080 obj:[0x7f39e876e000-0x7f39e876f000] threadId : 7f39e857f700
	:3:rocvirtual.cpp :2521: 43544495591 us: [7f39e857f700]! ShaderName : amd_rocclr_fillBuffer
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp	



### Other tips and tricks with rocgdb

#### Use i th to see a list of all active host threads. Currently viewing thread 1 (default)

	jychang48@jychang48-workstat	tion: ~/Downloads/hiptutorial/hip		Q = _ 0 😣
23				
24 25	<pre>const int id_l = id - 1; const int id r = id + 1;</pre>	:3:rocvirtual.cpp Z11NormKernel2iPKdPd	:2521: 43544718546 us: [7f39	e8446700]! ShaderName :
26	const int id d = id - stride;			
27	const int id u = id + stride;	:3:hip memory.cpp	:331 : 43544719486 us: 15157	5: [7f3a02b7c8c0] hipMemcpv:
28		Returned hipSuccess : : du		
29	AU[id] = (-U[id l] + 2*U[id] - U[id r])/(dx*dx) +	Iteration: 0 - Residual:		
30	(-U[id_d] + 2*U[id] - U[id_u])/(dy*dy);	:3:hip_device_runtime.cpp	:458 : 43544719507 us: 15157	5: [7f3a02b7c8c0] hipDeviceS
31 32	}	ynchronize ( )		
32	}		:470 : 43544719512 us: 15157	5: [7f3a02b7c8c0] hipDeviceS
33		ynchronize: Returned hipSuc		
34	<pre>void LocalLaplacian(grid_t&amp; grid, mesh_t&amp; mesh,</pre>	:3:hip_event.cpp	:310 : 43544719523 us: 15157	5: [/f3a02b/c8c0] hipEventRe
35 36 37	hipStream_t stream,	cord ( event:0x948d00, stre		
30	dfloat* d_U, dfloat* d AU) {	:3:hip_event.cpp cord: Returned hipSuccess :	:352 : 43544719530 us: 15157	5: [/I3a02b/C8C0] htpEventRe
38		Cord. Recurned htpsuccess .		
39	//there are (Nx-2)x(Ny-2) node on the interior of the mesh	Thread 1 "Jacobi hip" hit B	reakpoint 1, LocalLaplacian (	grid= mesh= stream=0
	-> int localNx = mesh.Nx-2;	x789710 d U=0x7f3893e00000	d All=0x7f388bc00000) at Lan	lacian cnn:40
41	<pre>int localNy = mesh.Ny-2;</pre>	(gdb) i th	· _ · ·	
42		Id Target Id	F	rame
43	<pre>int xthreads = 16;</pre>		0 (LWP 151575) "Jacobi_hip" L	
44 45	<pre>int ythreads = 16;</pre>		=0x7f3893e00000, d_AU=0x7f388	
45			0 (LWP 151591) "Jacobi_hip" 0	x00007f3a035b9aff in poll ()
46	<pre>dim3 threads(xthreads,ythreads,1);</pre>	from /lib/x86_64-linux-gnu		
47	<pre>dim3 blocks((localNx+xthreads-1)/xthreads,</pre>		0 (LWP 151592) "Jacobi_hip" 0	x00007f3a035c65ce in epoll_w
48	<pre>(localNy+ythreads-1)/ythreads, 1);</pre>	ait () from /lib/x86_64-lin		
49 50	hinlaunchKarnal(CL/LacalLanlacianKarnal	4 Inread 0x/T39e253e/0 ) from /lib/x86 64-linux-gn	0 (LWP 151593) "Jacobi_hip" 0	x0000/T3a035000 in locti (
51	hipLaunchKernelGGL(LocalLaplacianKernel, blocks,		0 (LWP 151595) "Jacobi hip" 0	$\sqrt{00007f}$
52	threads,		ib/x86 64-linux-gnu/libpthrea	
53	0, stream,		0 (LWP 151596) "Jacobi hip" 0	
54	localNx, localNy, mesh.Nx,		ib/x86 64-linux-gnu/libpthrea	
51 52 53 54 55 56	mesh.dx, mesh.dy,		0 (LWP 151597) "Jacobi hip" 0	
56	d U, d AU);		ib/x86 64-linux-gnu/libpthrea	
	e/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp	(gdb)		

### And more...

ROCgdb has several other features and capabilities not covered in this presentation. See the following for much more:

https://docs.amd.com/bundle/ROCDebugger-User-Guide-v5.2/page/index.html /opt/rocm-5.2.0/share/doc/rocgdb/rocannotate.pdf /opt/rocm-5.2.0/share/doc/rocgdb/rocgdb.pdf /opt/rocm-5.2.0/share/doc/rocgdb/rocrefcard.pdf /opt/rocm-5.2.0/share/doc/rocgdb/rocstabs.pdf

### Disclaimer

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED 'AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Third-party content is licensed to you directly by the third party that owns the content and is not licensed to you by AMD. ALL LINKED THIRD-PARTY CONTENT IS PROVIDED "AS IS" WITHOUT A WARRANTY OF ANY KIND. USE OF SUCH THIRD-PARTY CONTENT IS DONE AT YOUR SOLE DISCRETION AND UNDER NO CIRCUMSTANCES WILL AMD BE LIABLE TO YOU FOR ANY THIRD-PARTY CONTENT. YOU ASSUME ALL RISK AND ARE SOLELY RESPONSIBLE FOR ANY DAMAGES THAT MAY ARISE FROM YOUR USE OF THIRD-PARTY CONTENT.

© 2022 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, ROCm, Radeon, Radeon Instinct and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names are for informational purposes only and may be trademarks of their respective owners.

The OpenMP name and the OpenMP logo are registered trademarks of the OpenMP Architecture Review Board.

#