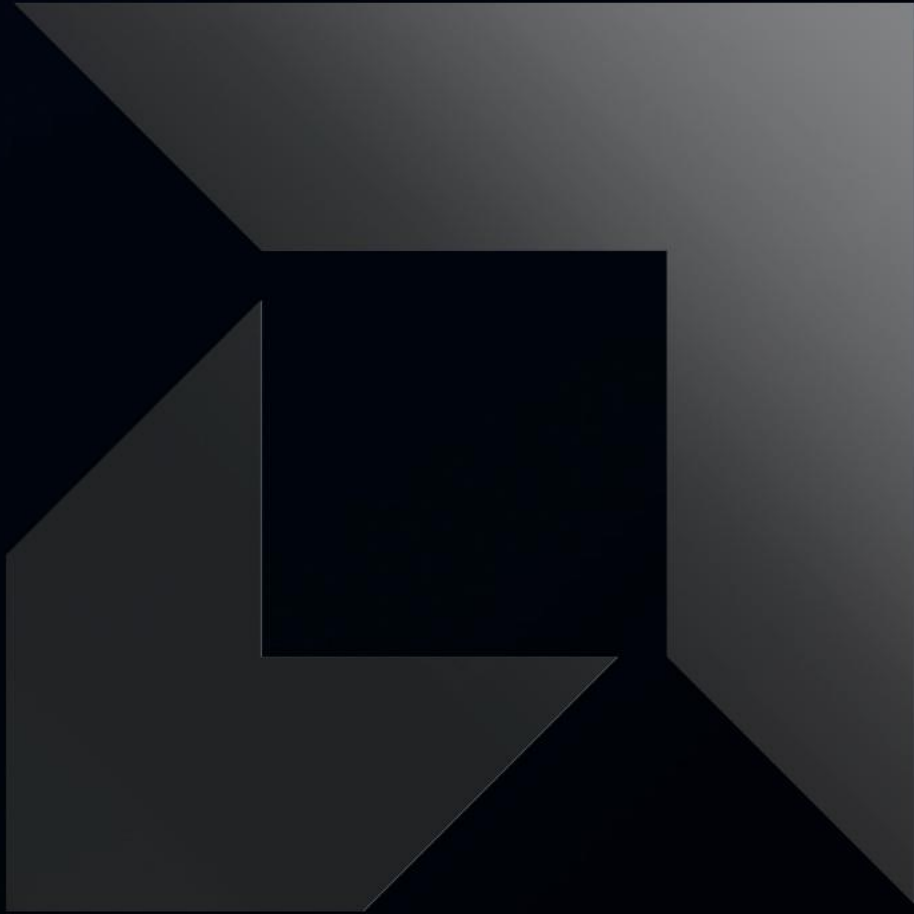




AMD Debugger: ROCgdb

Presenter: Bob Robey
Oct 21-23, 2025
AMD @ Tsukuba University

AMD 
together we advance_

What is covered in the Rocgdb Debugger Presentation

- Understand what information is accessible, particularly from the GPU, in the debugger
- Understand how to run the rocgdb debugger
- Learn what kind of interfaces are available for running the debugger

Rocgdb

- AMD ROCm™ source-level debugger for Linux®
- based on the GNU Debugger (GDB)
 - tracks upstream GDB master
 - standard GDB commands for both CPU and GPU debugging
- considered a prototype
 - focus on source line debugging
 - no symbolic variable debugging yet
- As GDB fork it can be used with other tools that use GDB as backend
- Exercises: <https://github.com/amd/HPCTrainingExamples/tree/main/Rocgdb>

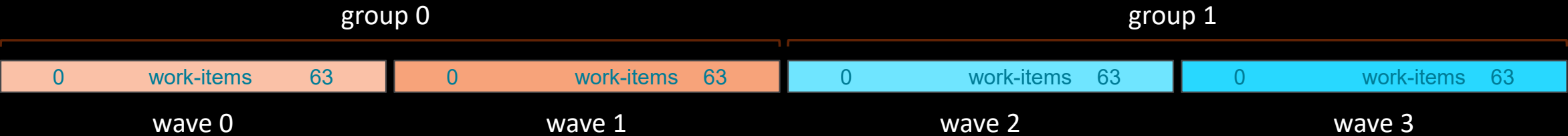
Simple saxpy kernel

```
1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n, d_x, 1, d_y, 1);
26     hipDeviceSynchronize();
27 }
28
```

classic saxpy operation
one array index = one work-item

size of arrays = 256

two groups
each 128 work-items



Cause a page fault

```

1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n, d_x, 1, d_y, 1);
26     hipDeviceSynchronize();
27 }
28

```

Could break it by forcing out of bounds read here by changing the index

Easier through commenting out the allocations.
(also possible to initialize the pointers to nullptr)

It's important to synchronize before exit.
Otherwise, the CPU thread may quit before the GPU gets a chance to report the error.

Compilation with hipcc

```

1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n,
26     hipDeviceSynchronize();
27 }
28

```

Need to set the target hardware

- gfx906 – MI50, MI60, Radeon™ 7
- gfx908 – MI100
- gfx90a – MI200
- gfx942 - MI300A

Can set multiple targets for different devices

```
saxpy$ hipcc --offload-arch=gfx90a -o saxpy saxpy.cpp
```

Execution

```
1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n,
26     hipDeviceSynchronize();
27 }
28
```

```
saxpy$ hipcc --offload-arch=gfx90a -o saxpy saxpy.cpp
saxpy$ ./saxpy
```

Get a page fault

```

1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n, d_x, d_y, a);
26     hipDeviceSynchronize();
27 }
28

```

```
saxpy$ hipcc --offload-arch=gfx90a -o saxpy saxpy.cpp
```

```
saxpy$ ./saxpy
```

```
Memory access fault by GPU node-2 (Agent handle: 0x2284d90) on address (nil). Reason: Unknown.
```

```
Aborted (core dumped)
```

```
saxpy$
```

And BOOM, here is our expected memory violation

Common gdb commands

Start GDB (GNU Debugger)

- **`gdb <program> [core dump]`**
- **`gdb -args <program> <args>`**
- **`gdb -help`**

Run commands

- `r[un]`** - Runs the program until a breakpoint or error
- `c[ontinue]`** - Continues running the program until the next breakpoint or error
- `q[uit]` or **`kill`** - Quits gdb**
- `fin[ish]`** - Runs until current function or loop is finished
- `n[ext]`** - Runs the next line of the program
 - `n N`** - Runs the next N lines of the program
- `s[tep]`** - Runs the next line of the program, stepping into any called routines
- `until N`** - Runs until you get N lines after the current line

Breakpoint commands

- `b[reakpoint] <where>`** – set breakpoint
 - `b main`** - Puts a breakpoint at the beginning of the program
 - `b`** - Puts a breakpoint at the current line
 - `b N`** - Puts a breakpoint at line N
 - `b +N`** - Puts a breakpoint N lines down from the current line
 - `b fn`** - Puts a breakpoint at the beginning of function "fn"

`b/w <where> if <condition>` – conditional breakpoint or watch

`i[nfo] b[reak]` - list breakpoints

`dis[able] N` - disable breakpoint number N

`en[able] N` – enables breakpoint number N

`d[ele] N` – delete breakpoint number N

`clear` – clear all breakpoints

Print commands

`[h]elp <command>`

`[p]rint var` - Prints the current value of the variable "var"

`[l]ist` – list lines

`bt (backtrace)` - Prints a stack trace

Movement

`up` - Goes up a level in the stack

`[do]wn` - Goes down a level in the stack

Execution with rocdbg

```

1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n, d_x, d_y, 1, 1);
26     hipDeviceSynchronize();
27 }
28

```

```
saxpy$ rocdbg saxpy
```

Remember to use rocdbg `--args` when passing arguments to program being debugged

Get more information

```

1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n,
26     hipDeviceSynchronize();
27 }
28

```

Reports segmentation fault in the saxpy kernel.

```

(gdb) run
Starting program: /home/gmarkoma/saxpy
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
[New Thread 0x7ffffd428700 (LWP 10456)]
Warning: precise memory violation signal reporting is not enabled, reported
location may not be accurate. See "show amdgpu precise-memory".

Thread 3 "saxpy" received signal SIGSEGV, Segmentation fault.
[Switching to thread 3, lane 0 (AMDGPU Lane 1:2:1:1/0 (0,0,0)[0,0,0])]
0x00007ffff7ec1094 in saxpy(int, float const*, int, float*, int) () from file:///home/gmarkoma/s
axpy#offset=8192&size=13832
(gdb)

```

Compile with -ggdb

```
1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n,
26     hipDeviceSynchronize());
27 }
28
```

```
saxpy$ hipcc -ggdb --offload-arch=gfx90a -o saxpy saxpy.cpp
```

Get more details

```

1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n, d_x, d_y);
26     hipDeviceSynchronize();
27 }
28

```

more details

- what kernel
- what file:line

```

(gdb) run
Starting program: /home/gmarkoma/saxpy
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
[New Thread 0x7ffff7c10000 (LWP 10637)]
Warning: precise memory violation signal reporting is not enabled, reported
location may not be accurate. See "show amdgpu precise-memory".

Thread 3 "saxpy" received signal SIGSEGV, Segmentation fault.
[Switching to thread 3, lane 0 (AMDGPU Lane 1:2:1:1/0 (0,0,0)[0,0,0])]
0x00007ffff7c10000 in saxpy () at saxpy.cpp:10
10      y[i] += a*x[i];
(gdb)

```

But where's my stack trace?

To get exceptions reported precisely: set `amdgpu precise-memory` on

List threads

```

1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n,
26     hipDeviceSynchronize();
27 }
28

```

What segfaulted is a GPU wave.
It does not have your CPU stack.
List threads to see what's going on.

```

(gdb) i th
      Id      Target Id      Frame
1      Thread 0x7ffff7fe6e80 (LWP 10633) "saxpy" 0x00007ffffee0fc499 in rocr::core::InterruptSign
al::WaitRelaxed(hsa_signal_condition_t, long, unsigned long, hsa_wait_state_t) ()
    from /opt/rocm-5.2.0/lib/libhsa-runtime64.so.1
2      Thread 0x7ffff428700 (LWP 10637) "saxpy" 0x00007ffff5e1972b in ioctl ()
    from /lib64/libc.so.6
* 3      AMDGPU Wave 1:2:1:1 (0,0,0)/0 "saxpy" 0x00007ffff7ec1094 in saxpy () at saxpy.cpp:10
4      AMDGPU Wave 1:2:1:2 (0,0,0)/1 "saxpy" 0x00007ffff7ec1094 in saxpy () at saxpy.cpp:10
5      AMDGPU Wave 1:2:1:3 (1,0,0)/0 "saxpy" 0x00007ffff7ec1094 in saxpy () at saxpy.cpp:10
6      AMDGPU Wave 1:2:1:4 (1,0,0)/1 "saxpy" 0x00007ffff7ec1094 in saxpy () at saxpy.cpp:10
(gdb)

```

Switch to the CPU thread

```

1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n,
26     hipDeviceSynchronize();
27 }
28

```

t 1
(thread 1)
It's in the HSA runtime.

```

(gdb) t 1
[Switching to thread 1 (Thread 0x7ffff7fe6e80 (LWP 10633))]
#0  0x00007ffffee0fc499 in rocr::core::InterruptSignal::WaitRelaxed(hsa_signal_condition_t, long,
    unsigned long, hsa_wait_state_t) () from /opt/rocm-5.2.0/lib/libhsa-runtime64.so...
(gdb)

```

But how did it get there?

See the stack trace of the CPU thread

```
1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n, d_x, d_y);
26     hipDeviceSynchronize();
27 }
28
```

HSA runtime

HIP runtime

The CPU thread is currently waiting on the device to finish ☺

```
(gdb) where
#0  0x00007ffffee0fc499 in rocr::core::InterruptSignal::WaitRelaxed(hsa_signal_condition_t, long, unsigned long, hsa_wait_state_t) () from /opt/rocm-5.2.0/lib/libhsa-runtime64.so.1
#1  0x00007ffffee0fc36a in rocr::core::InterruptSignal::WaitAcquire(hsa_signal_condition_t, long, unsigned long, hsa_wait_state_t) () from /opt/rocm-5.2.0/lib/libhsa-runtime64.so.1
#2  0x00007ffffee0f0869 in rocr::HSA::hsa_signal_wait_scacquire(hsa_signal_s, hsa_signal_condition_t, long, unsigned long, hsa_wait_state_t) () from /opt/rocm-5.2.0/lib/libhsa-runtime64.so.1
#3  0x00007ffff67bdd43 in bool roc::WaitForSignal<false>(hsa_signal_s, bool) () from /opt/rocm-5.2.0/lib/libamdhip64.so.5
#4  0x00007ffff67b5836 in roc::VirtualGPU::HwQueueTracker::CpuWaitForSignal(roc::ProfilingSignal*) () from /opt/rocm-5.2.0/lib/libamdhip64.so.5
#5  0x00007ffff67b77cf in roc::VirtualGPU::releaseGpuMemoryFence(bool) () from /opt/rocm-5.2.0/lib/libamdhip64.so.5
#6  0x00007ffff67b9523 in roc::VirtualGPU::flush(amd::Command*, bool) () from /opt/rocm-5.2.0/lib/libamdhip64.so.5
#7  0x00007ffff67b9db0 in roc::VirtualGPU::submitMarker(amd::Marker&) () from /opt/rocm-5.2.0/lib/libamdhip64.so.5
#8  0x00007ffff678ec2e in amd::Command::enqueue() () from /opt/rocm-5.2.0/lib/libamdhip64.so.5
#9  0x00007ffff678f1e0 in amd::Event::notifyCmdQueue(bool) () from /opt/rocm-5.2.0/lib/libamdhip64.so.5
#10 0x00007ffff678f28c in amd::Event::awaitCompletion() () from /opt/rocm-5.2.0/lib/libamdhip64.so.5
#11 0x00007ffff6791fdc in amd::HostQueue::finish() () from /opt/rocm-5.2.0/lib/libamdhip64.so.5
#12 0x00007ffff65c25f9 in hipDeviceSynchronize () from /opt/rocm-5.2.0/lib/libamdhip64.so.5
#13 0x000000000020d615 in main () at saxpy.cpp:25
(gdb)
```

Quick tip

- CPUs in AAC MI300A nodes have 96 cores / 192 threads.
- If you're debugging an app with OpenMP® threading and OMP_NUM_THREADS is not set you will see 192 CPU threads in rocdbg.
- Set OMP_NUM_THREADS=1 when debugging GPU codes that don't have CPU specific OpenMP® regions
- For debugging with MPI, attach rocdbg to individual MPI processes

Breakpoints – Common pitfalls

We try to put a breakpoint in line 22 but it is declared as line 24.

```
--Type <RET> for more, q to quit, c to continue without paging--For help, type "help".  
Type "apropos word" to search for commands related to "word"..  
Reading symbols from saxpy..  
(gdb) b saxpy.cpp:22  
Breakpoint 1 at 0x20d57f: file saxpy.cpp, line 24.  
(gdb) _
```

Simple saxpy kernel – Where is our code?

```
1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, int incx, float* y, int incy)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n, d_x, 1, d_y, 1);
26     hipDeviceSynchronize();
27 }
28
```

Breakpoints – If possible, debug with optimization turned off

We try to put a breakpoint in line 22 but it is declared as line 24.

```
--Type <RET> for more, q to quit, c to continue without paging--For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from saxpy...
(gdb) b saxpy.cpp:22
Breakpoint 1 at 0x20d57f: file saxpy.cpp, line 24.
(gdb) _
```

Default compiler optimization for hipcc is –O3, compile with –O0

```
saxpy$ hipcc -ggdb -O0 --offload-arch=gfx90a -o saxpy saxpy.cpp
```

Creating a breakpoint again and it is declared in the correct line

```
--Type <RET> for more, q to quit, c to continue without paging--For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from saxpy...
(gdb) b saxpy.cpp:22
Breakpoint 1 at 0x219dec: file saxpy.cpp, line 22.
(gdb) _
```

Running and architecture

Running with the keystroke *r* and stops at the breakpoint

```
--Type <RET> for more, q to quit, c to continue without paging--For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from saxpy...
(gdb) b saxpy.cpp:22
Breakpoint 1 at 0x219dec: file saxpy.cpp, line 22.
(gdb) r
Starting program: /home/gmarkoma/saxpy
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
[New Thread 0x7ffffd428700 (LWP 16916)]

Thread 1 "saxpy" hit Breakpoint 1, main () at saxpy.cpp:22
(gdb) _
```

More information about the thread with the command *i th*

```
(gdb) i th
   Id   Target Id                                     Frame
*  1   Thread 0x7ffff7fe6e80 (LWP 16912) "saxpy" main () at saxpy.cpp:22
   2   Thread 0x7ffffd428700 (LWP 16916) "saxpy" 0x00007ffff5e1972b in ioctl () from /lib64/libc.so.6
(gdb) _
```

We can see on what device is the thread with the ***show architecture*** command

```
(gdb) show architecture
The target architecture is set to "auto" (currently "i386:x86-64").
(gdb)
```

Breakpoint kernel and architecture

Breakpoint on the kernel called saxpy with the command **b saxpy**

```
(gdb) b saxpy
Function "saxpy" not defined.
Make breakpoint pending on future shared library load? (y or [n]) yBreakpoint 2 (saxpy) pending.
(gdb)
```

You can continue with the command **c**

```
(gdb) c
Continuing.
[New Thread 0x7ffffdefff700 (LWP 16937)]
[New Thread 0x7ffffecaff700 (LWP 16938)]
[Thread 0x7ffffdefff700 (LWP 16937) exited]
[Switching to thread 5, lane 0 (AMDGPU Lane 1:2:1:1/0 (0,0,0)[0,0,0])]

Thread 5 "saxpy" hit Breakpoint 2, with lanes [0-63], saxpy (n=256, x=0x7ffffec700000, incx=1, y=0x7ffffec701000, incy=1) at saxpy.cpp:9
```

We can see on what device is the thread with the command **show architecture**

```
(gdb) show architecture
The target architecture is set to "auto" (currently "amdgc:gfx90a").
```

“GUIs”

rocgdb -tui saxpy

saxpy.cpp

```
1 #include "hip/hip_runtime.h"
2 #include <stdio.h>
3
4 __constant__ float a = 1.0f;
5
6 __global__
7 void saxpy(int n, float const* x, int incx, float* y, int incy)
8 {
9     int i = blockIdx.x*blockDim.x + threadIdx.x;
10    if (i < n) y[i] = a*x[i] + y[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
17
18     float *d_x, *d_y;
19     //hipMalloc(&d_x, size);
20     //hipMalloc(&d_y, size);
21
22     int num_groups= 2;
23     int group_size=128;
24     saxpy<<<num_groups,group_size>>>(n, d_x, 1, d_y, 1);
25 }
```

amd-dbgapi AMDGPU Wave 1:2:1:1 In: saxpy

Type "apropos word" to search for commands related to "word"...

Reading symbols from saxpy...

(gdb) run

Starting program: /home/gmarkoma/saxpy

[Thread debugging using libthread_db enabled]

Using host libthread_db library "/lib64/libthread_db.so.1".

[New Thread 0x7ffff428700 (LWP 11074)]

Warning: precise memory violation signal reporting is not enabled, reported location may not be accurate. See "show amdgpu precise-memory".

Thread 3 "saxpy" received signal SIGSEGV, Segmentation fault.

[Switching to thread 3, lane 0 (AMDGPU Lane 1:2:1:1/0 (0,0,0)[0,0,0])]

0x00007ffff428700 in saxpy () at saxpy.cpp:10

(gdb)

cgdb -d rocgdb saxpy

saxpy: cgdb — Konsole

File Edit View Bookmarks Settings Help

```
1 #include <hip/hip_runtime.h>
2
3 __constant__ float a = 1.0f;
4
5 __global__
6 void saxpy(int n, float const* x, int incx, float* y, int incy)
7 {
8     int i = blockIdx.x*blockDim.x + threadIdx.x;
9     if (i < n)
10        y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n, d_x, 1, d_y, 1);
26     hipDeviceSynchronize();
27 }
```

/mnt/shared/codes/saxpy/saxpy.hip.cpp

[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) █

GDB Dashboard

- To show all the debugging information in one screen, you can also use GDB dashboard. Works with rocgdb.
- Shows:
 - Assembly
 - Breakpoints
 - Registers
 - Stack
 - Threads
 - Variables
 - ... and more
- <https://github.com/cyrus-and/gdb-dashboard>

GDB dashboard

Output/messages

17 for (i = 0; i < text_length; i++) {

Assembly

0x0000555555551ec 48 8b 45 f8 encrypt+103 mov rax,QWORD PTR [rbp-0x8]
0x0000555555551f0 48 01 d0 encrypt+107 add rax,rdx
0x0000555555551f3 31 ce encrypt+110 xor esi,ecx
0x0000555555551f5 89 f2 encrypt+112 mov edx,esi
0x0000555555551f7 88 10 encrypt+114 mov BYTE PTR [rax],dl
0x0000555555551f9 48 83 45 f8 01 encrypt+116 add QWORD PTR [rbp-0x8],0x1
0x0000555555551fe 48 8b 45 f8 encrypt+121 mov rax,QWORD PTR [rbp-0x8]
0x000055555555202 48 3b 45 e8 encrypt+125 cmp rax,QWORD PTR [rbp-0x18]
0x000055555555206 72 bb encrypt+129 jb 0x555555551c3 <encrypt+62>
0x000055555555208 90 encrypt+131 nop

Breakpoints

[1] break at 0x0000555555552d9 in xor.c:56 for xor.c:56 hit 1 time
[2] break at 0x000055555555199 in xor.c:13 for encrypt hit 1 time
[3] break at 0x00005555555521b in xor.c:27 for dump if i = 5
[4] write watch for output[10] hit 1 time

Expressions

[1] text[i] = 32 ' '
[2] password[i % password_length] = 101 'e'
[3] output[i] = 69 'E'

History

\$\$1 = 0x555555559260 "\f\032\v\006\022\004\032\001\037E": 12 '\f'
\$\$0 = 0x7fffffffef2c "hunter2": 104 'h'

Memory

password
0x00007fffffffef2c 68 75 6e 74 65 72 32 00 64 6f 65 73 6e 74 20 6c hunter2·doesnt·l
text
0x00007fffffffef34 64 6f 65 73 6e 74 20 6c 6f 6f 6b 20 6c 69 6b 65 doesnt·look·like
0x00007fffffffef44 20 73 74 61 72 73 20 74 6f 20 6d 65 00 48 4f 53 ·stars·to·me·HOS
output
0x0000555555559260 0c 1a 0b 07 0b 06 12 04 1a 01 1f 45 00 00 00 00E....
0x0000555555559270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Registers

rax 0x000055555555926b rbx 0x0000000000000000 rcx 0x0000000000000065
rdx 0x0000000000000045 rsi 0x0000000000000045 rdi 0x00007fffffffef40
rbp 0x00007fffffffef20 rsp 0x00007ffffffeb0 r8 0x0000000000000003
r9 0x0000000000a31d0 r10 0x0000555555559010 r11 0x0000000000000030
r12 0x0000555555550a0 r13 0x00007ffffffed40 r14 0x0000000000000000
r15 0x0000000000000000 rip 0x0000555555551f9 eflags [IF]
cs 0x00000033 ss 0x0000002b ds 0x00000000
es 0x00000000 fs 0x00000000 gs 0x00000000

Source

12 /* obtain the lengths */
13 password_length = strlen(password);
14 text_length = strlen(text);
15
16 /* perform the encryption */
17 for (i = 0; i < text_length; i++) {
18 output[i] = text[i] ^ password[i % password_length];
19 }
20 }
21

Stack

[0] from 0x0000555555551f9 in encrypt+116 at xor.c:17
[1] from 0x0000555555552f0 in main+139 at xor.c:56

Threads

[1] id 9 name xor from 0x0000555555551f9 in encrypt+116 at xor.c:17

Variables

arg password = 0x7fffffffef2c "hunter2": 104 'h'
arg text = 0x7fffffffef34 "doesnt look like stars to me": 100 'd'
arg output = 0x555555559260 "\f\032\v\006\022\004\032\001\037E": 12 '\f'
loc password_length = 7
loc text_length = 28
loc i = 11

>>> |

rocgdb + gdbgui

breakpoint in CPU code



Load Binary /mnt/shared/codes/saxpy/saxpy

show filesystem fetch disassembly reload file jump to line /mnt/shared/codes/saxpy/saxpy.hip.cpp:22 (27 lines total)

```
1  #include <hip/hip_runtime.h>
2
3  __constant__ float a = 1.0f;
4
5  __global__
6  void saxpy(int n, float const* x, float* y)
7  {
8      int i = blockDim.x*blockIdx.x + threadIdx.x;
9      if (i < n)
10         y[i] += a*x[i];
11 }
12
13 int main()
14 {
15     int n = 256;
16     std::size_t size = sizeof(float)*n;
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     int group_size = 128;
25     saxpy<<<num_groups, group_size>>>(n, d_x, d_y);
26 }
27
(end of file)
```

running command: /opt/rocm/bin/rocgdb

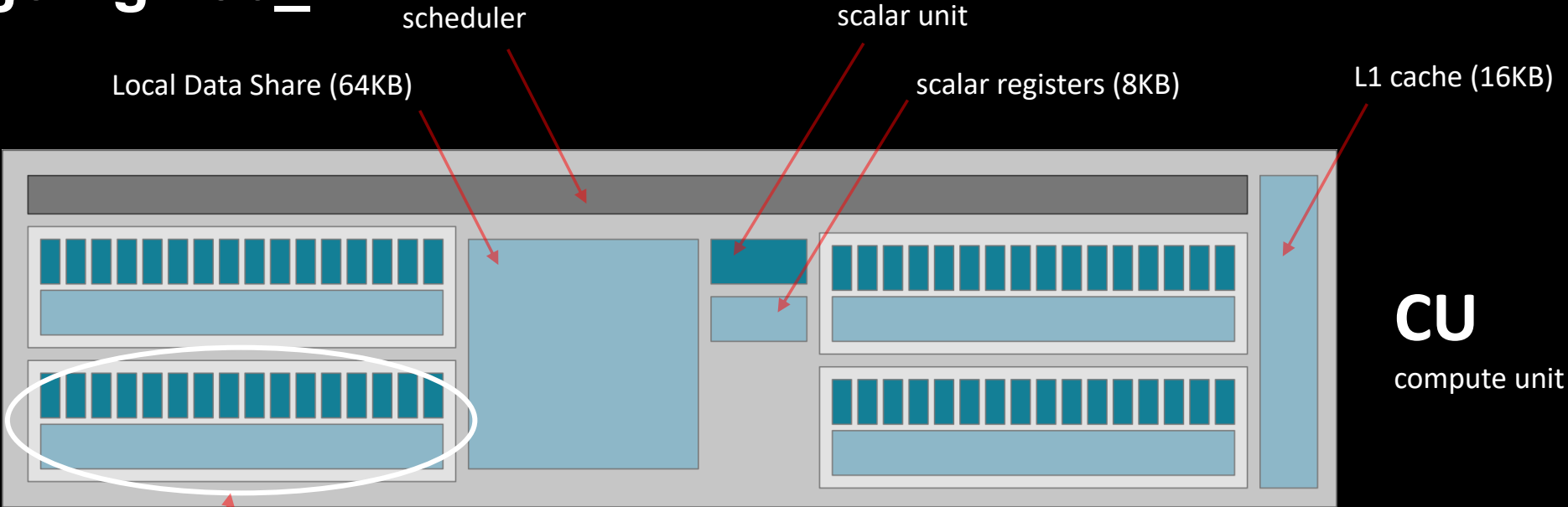
GNU gdb (rocm-rel-4.5-56) 11.1
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <<http://gnu.org/licenses/gpl.html>>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" 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.
New UI allocated
(gdb) █

source

console

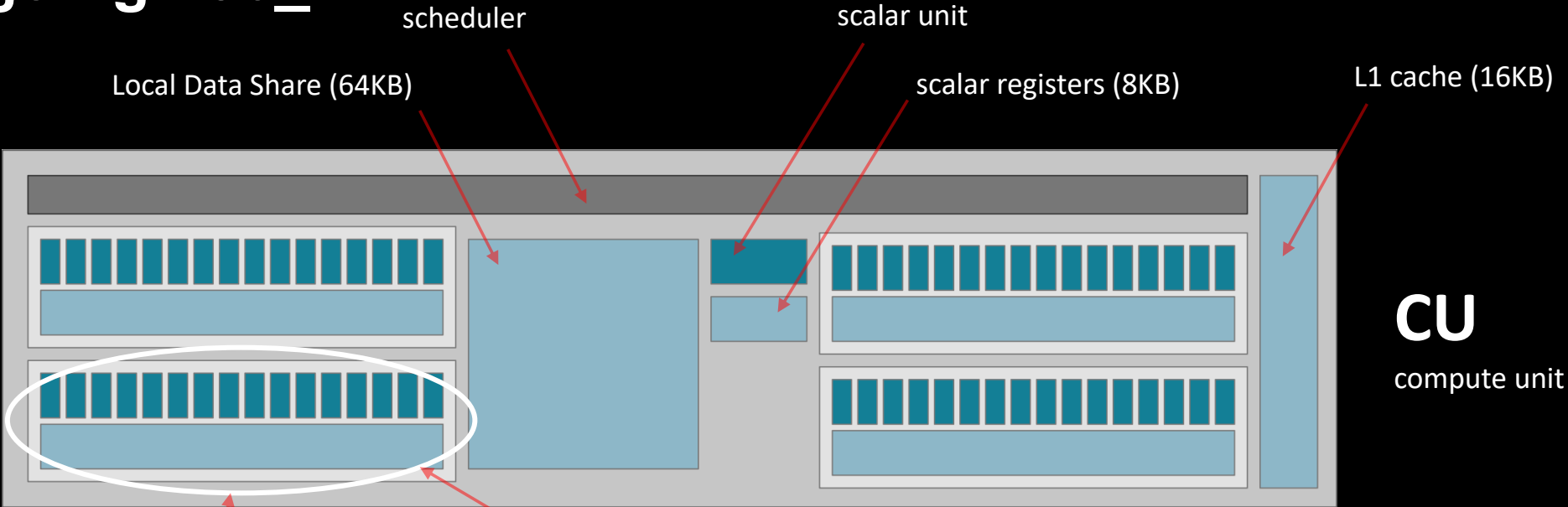
amdgcn:gfx90_



typically described as

- a 16-way SIMD unit
- with 64KB of registers

amdgcn:gfx90_



CU
compute unit

- typically described as
- a 16-way SIMD unit
 - with 64KB of registers

- from the standpoint of rocGDB
- a **core**
 - executing up to 10 **threads**
 - with vector length of 64 **lanes**
 - and containing 256 vector **registers**

List threads / waves

i th
(info threads)
some CPU threads

4 GPU “threads” (waves)

(gdb) i th			
Id	Target Id	Frame	
1	Thread 0x7ffff7fe6e80 (LWP 16912) "saxpy"	0x00007ffffe0fc4c0 in rocr::core::InterruptSignal: t) () from /opt/rocm-5.2.0/lib/libhsa-runtime64.so.1	
2	Thread 0x7ffffed428700 (LWP 16916) "saxpy"	0x00007ffff5e1972b in ioctl1 () from /lib64/libc.so	
4	Thread 0x7ffffecaff700 (LWP 16938) "saxpy"	0x00007ffffe0fc4af in rocr::core::InterruptSignal: t) () from /opt/rocm-5.2.0/lib/libhsa-runtime64.so.1	
* 5	AMDGPU Wave 1:2:1:1 (0,0,0)/0 "saxpy"	saxpy (n=256, x=0x7ffffec700000, incx=1, y=0x7ffffec700000)	
6	AMDGPU Wave 1:2:1:2 (0,0,0)/1 "saxpy"	saxpy (n=256, x=0x7ffffec700000, incx=1, y=0x7ffffec700000)	
7	AMDGPU Wave 1:2:1:3 (1,0,0)/0 "saxpy"	saxpy (n=256, x=0x7ffffec700000, incx=1, y=0x7ffffec700000)	
8	AMDGPU Wave 1:2:1:4 (1,0,0)/1 "saxpy"	saxpy (n=256, x=0x7ffffec700000, incx=1, y=0x7ffffec700000)	

Wave details

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

```
(gdb) i th
Id      Target Id      Frame
1      Thread 0x7ffff7fe6e80 (LWP 16912) "saxpy" 0x00007ffffee0fc4c0 in rocr::core::InterruptSignal:
t) () from /opt/rocm-5.2.0/lib/libhsa-runtime64.so.1
2      Thread 0x7ffffd428700 (LWP 16916) "saxpy" 0x00007ffff5e1972b in ioctl () from /lib64/libc.so
4      Thread 0x7ffffecaff700 (LWP 16938) "saxpy" 0x00007ffffee0fc4af in rocr::core::InterruptSignal:
t) () from /opt/rocm-5.2.0/lib/libhsa-runtime64.so.1
* 5      AMDGPU Wave 1:2:1:1 (0,0,0)/0 "saxpy" saxpy (n=256, x=0x7ffffec700000, incx=1, y=0x7ffffec700000)
6      AMDGPU Wave 1:2:1:2 (0,0,0)/1 "saxpy" saxpy (n=256, x=0x7ffffec700000, incx=1, y=0x7ffffec700000)
7      AMDGPU Wave 1:2:1:3 (1,0,0)/0 "saxpy" saxpy (n=256, x=0x7ffffec700000, incx=1, y=0x7ffffec700000)
8      AMDGPU Wave 1:2:1:4 (1,0,0)/1 "saxpy" saxpy (n=256, x=0x7ffffec700000, incx=1, y=0x7ffffec700000)
```

agent (GPU) ID

(HSA) queue ID

dispatch number

wave ID

workgroup
(x, y, z)

wave ID
(within group)

More advanced things you can do

- inspect / modify registers
- inspect / modify memory
- inspect / modify LDS
- step through the assembly one instruction at a time
- Check race conditions by stepping code in separate GPU waves

List agents

info agents
➤ shows devices + properties

(gdb) info agents

	Id	State	Target	Id	Architecture	Device Name	Cores	Threads	Location
*	1	A	AMDGPU Agent	(GPUID 31957)	gfx90a	aldebaran	440	3520	29:00.0

gfx90a
MI200 series

SIMDs
(CUs x 4)

max waves
(SIMDs x 8)

List queues

info queues
➤ shows HSA queues

(gdb) info queues

Id	Target Id	Type	Read	Write	Size	Address
1	AMDGPU Queue 1:1 (QID 0)	HSA (Multi)	4	4	4096	0x00007ffff7eb6000
* 2	AMDGPU Queue 1:2 (QID 1)	HSA (Multi)	0	2	262144	0x00007ffff7e40000

(gdb)

agent ID

queue ID

(AQL) packets read

(AQL) packets written

Dispatch details

info dispatches
➤ shows kernel dispatches

(gdb) info dispatches

Id	Target Id	Grid	Workgroup	Fence	Kernel Function
* 1	AMDGPU Dispatch 1:2:1 (PKID 0)	[256,1,1]	[128,1,1]	B Aa Ra	saxpy(int, float const*, int, float*, int)

agent ID

queue ID

dispatch ID

grid dimensions

group dimensions

kernel

Use AMD_LOG_LEVEL=3 to find errors without having to use the debugger

```

:3:devprogram.cpp      :2978: 157529658660 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: _Z5saxpyiPKfiPfi
:3:hip_module.cpp      :365 : 157529658684 us: 224178: [tid:0x7f59c7439e80] ihipModuleLaunchKernel ( 0x0x12e9720, 256, 1, 1, 128, 1, 1, 0, stream:<null>, 0x7fff94e2e07
0, char array:<null>, event:0, 0, 0 )
:3:rocdevice.cpp       :2686: 157529658695 us: 224178: [tid:0x7f59c7439e80] number of allocated hardware queues with low priority: 0, with normal priority: 0, with hig
h priority: 0, maximum per priority is: 4
:3:rocdevice.cpp       :2757: 157529663975 us: 224178: [tid:0x7f59c7439e80] created hardware queue 0x7f59c72f4000 with size 4096 with priority 1, cooperative: 0
:3:devprogram.cpp      :2675: 157529852150 us: 224178: [tid:0x7f59c7439e80] Using Code Object V4.
:3:devprogram.cpp      :2978: 157529853058 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_fillImage
:3:devprogram.cpp      :2978: 157529853065 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_fillBufferAligned2D
:3:devprogram.cpp      :2978: 157529853070 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_fillBufferAligned
:3:devprogram.cpp      :2978: 157529853076 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_copyImage1DA
:3:devprogram.cpp      :2978: 157529853080 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_copyBufferAligned
:3:devprogram.cpp      :2978: 157529853084 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_streamOpsWait
:3:devprogram.cpp      :2978: 157529853087 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_copyBuffer
:3:devprogram.cpp      :2978: 157529853091 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_streamOpsWrite
:3:devprogram.cpp      :2978: 157529853094 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_copyBufferRectAligned
:3:devprogram.cpp      :2978: 157529853096 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_gwsInit
:3:devprogram.cpp      :2978: 157529853099 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_copyBufferRect
:3:devprogram.cpp      :2978: 157529853101 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_copyImageToBuffer
:3:devprogram.cpp      :2978: 157529853105 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_copyBufferToImage
:3:devprogram.cpp      :2978: 157529853108 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: __amd_rocclr_copyImage
:3:rocvirtual.cpp      :753 : 157529853195 us: 224178: [tid:0x7f59c7439e80] Arg0:  = val:256
:3:rocvirtual.cpp      :679 : 157529853200 us: 224178: [tid:0x7f59c7439e80] Arg1:  = ptr:0x7f59bbb00000 obj:[0x7f59bbb00000-0x7f59bbb00400]
:3:rocvirtual.cpp      :753 : 157529853205 us: 224178: [tid:0x7f59c7439e80] Arg2:  = val:1
:3:rocvirtual.cpp      :679 : 157529853209 us: 224178: [tid:0x7f59c7439e80] Arg3:  = ptr:0x7f59bbb01000 obj:[0x7f59bbb01000-0x7f59bbb01400]
:3:rocvirtual.cpp      :753 : 157529853213 us: 224178: [tid:0x7f59c7439e80] Arg4:  = val:1
:3:rocvirtual.cpp      :2723: 157529853216 us: 224178: [tid:0x7f59c7439e80] ShaderName : _Z5saxpyiPKfiPfi
:3:hip_platform.cpp    :676 : 157529853233 us: 224178: [tid:0x7f59c7439e80] ihipLaunchKernel: Returned hipSuccess :
:3:hip_module.cpp      :509 : 157529853237 us: 224178: [tid:0x7f59c7439e80] hipLaunchKernel: Returned hipSuccess :
:3:hip_device_runtime.cpp :476 : 157529853243 us: 224178: [tid:0x7f59c7439e80] hipDeviceSynchronize ( )
:3:rocdevice.cpp       :2636: 157529853248 us: 224178: [tid:0x7f59c7439e80] No HW event
:3:rocvirtual.hpp      :62 : 157529853255 us: 224178: [tid:0x7f59c7439e80] Host active wait for Signal = (0x7f59c7442600) for -1 ns
:3:hip_device_runtime.cpp :488 : 157529853267 us: 224178: [tid:0x7f59c7439e80] hipDeviceSynchronize: Returned hipSuccess :
:3:hip_memory.cpp      :536 : 157529853279 us: 224178: [tid:0x7f59c7439e80] hipFree ( 0x7f59bbb00000 )
:3:rocdevice.cpp       :2093: 157529853291 us: 224178: [tid:0x7f59c7439e80] device=0x12d34f0, freeMem_ = 0xfeffffc00
:3:hip_memory.cpp      :538 : 157529853296 us: 224178: [tid:0x7f59c7439e80] hipFree: Returned hipSuccess :
:3:hip_memory.cpp      :536 : 157529853300 us: 224178: [tid:0x7f59c7439e80] hipFree ( 0x7f59bbb01000 )
:3:rocdevice.cpp       :2093: 157529853306 us: 224178: [tid:0x7f59c7439e80] device=0x12d34f0, freeMem_ = 0xff000000
:3:hip_memory.cpp      :538 : 157529853310 us: 224178: [tid:0x7f59c7439e80] hipFree: Returned hipSuccess :
:3:devprogram.cpp      :2978: 157529853333 us: 224178: [tid:0x7f59c7439e80] For Init/Fini: Kernel Name: _Z5saxpyiPKfiPfi

```

How to use rocgdb + gdbgui + Chrome

test if X forwarding works

```
ssh -X USERNAME@server
ssh -X login1._____.olcf.ornl.gov
srun -A VEN113 -N 1 -n 1 -c 64 --x11 --pty bash
xmessage -center hello!
```

install gdbgui

```
python3 -m pip install --user pipx
python3 -m userpath append ~/.local/bin
pipx install gdbgui
```

install Chrome

- Go to <https://www.google.com/chrome/>
- Click *Download Chrome*
- Click *64 bit .rpm (For Fedora/openSUSE)*
- Click *Accept and Install*

```
scp google-chrome-stable_current_x86_64.rpm USERNAME@home.ccs.ornl.gov:
ssh -X USERNAME@home.ccs.ornl.gov
mkdir ~/chrome
cd ~/chrome
rpm2cpio ../google-chrome-stable_current_x86_64.rpm | cpio -id
```

run rocgdb with gdbgui in Chrome

```
ssh -X USERNAME@home.ccs.ornl.gov
ssh -X login1._____.olcf.ornl.gov
srun -A VEN113 -N 1 -n 1 -c 64 --x11 --pty bash
gdbgui -g /opt/rocm/bin/rocgdb --no-browser &
~/chrome/opt/google/chrome/google-chrome 2>/dev/null &
```

- In Chrome, go to: <http://127.0.0.1:5000>
- Click *Load Binary* to load your binary (compiled with -ggdb)
- Step into a kernel
- Click *fetch disassembly*

```
show architecture
info threads
info queues
info dispatches
info registers
info reg vcc
info reg exec
s
si
n
ni
...
```

More resources for rocgdb

- /opt/rocm<-version>/share/doc/rocgdb/
 - rocgdb.pdf -- has additions for GPU commands
 - rocrefcard.pdf -- standard gdb reference card
- Presentations
 - https://www.olcf.ornl.gov/wp-content/uploads/2021/04/rocgdb_hipmath_ornl_2021_v2.pdf -- Justin Chang (AMD)
 - <https://lpc.events/event/11/contributions/997/attachments/928/1828/LPC2021-rocgdbdemo.pdf> – Andrew Stubbs (Siemens®) – See <https://youtu.be/IGWFph4SlpU> for 24 min video from presentation of debugging GCC offloading code (OpenACC and OpenMP®)

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 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.

© 2025 Advanced Micro Devices, Inc. All rights reserved.

AMD, the AMD Arrow logo, AMD ROCm, Radeon and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

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

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Siemens® is a registered trademark of Siemens Product Lifecycle Management Software Inc., or its subsidiaries or affiliates, in the United States and in other countries

