HIP Training Workshop – Day 3

ROCgdb & HIP math libraries

Justin Chang

Part 1: Intro to ROCgdb

Getting started

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.

So... cuda-gdb? Yes, and mostly no -- rocgdb *is* (or will be) gdb, that is it tracks upstream GDB master.

What can it do?

In addition to your usual host-debugging capabilities, a very brief overview of rocgdb's current functionality:

- Switching between and seeing info about wavefronts
- Read/write to hardware registers, global memory, and LDS/scratch
- Breakpoints
- Watchpoints
- ISA-level debugging, mapping of ISA to source lines

Before we can take a test drive, let's talk about the code we're going to use it on.

Jacobi Example

Our example code:

- Solves a Laplacian problem
- 5-point finite difference stencil
- May be distributed over multiple MPI ranks
- Performs several iteration of a Jacobi method

Laplacian equation

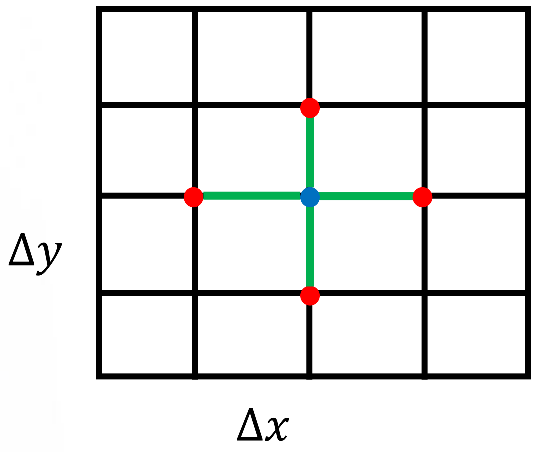
Laplace equation:

$$-\nabla^2 q = f$$

In two dimensions, with a 5-point finite difference stencil we discretize as:

$$\frac{-q_{i-1,j} + 2q_{i,j} - q_{i+1,j}}{\Delta x^2} + \frac{-q_{i,j-1} + 2q_{i,j} - q_{i,j+1}}{\Delta y^2} = f_{i,j}$$

On a rectangular lattice one point in the x–y plane:



Jacobi Example

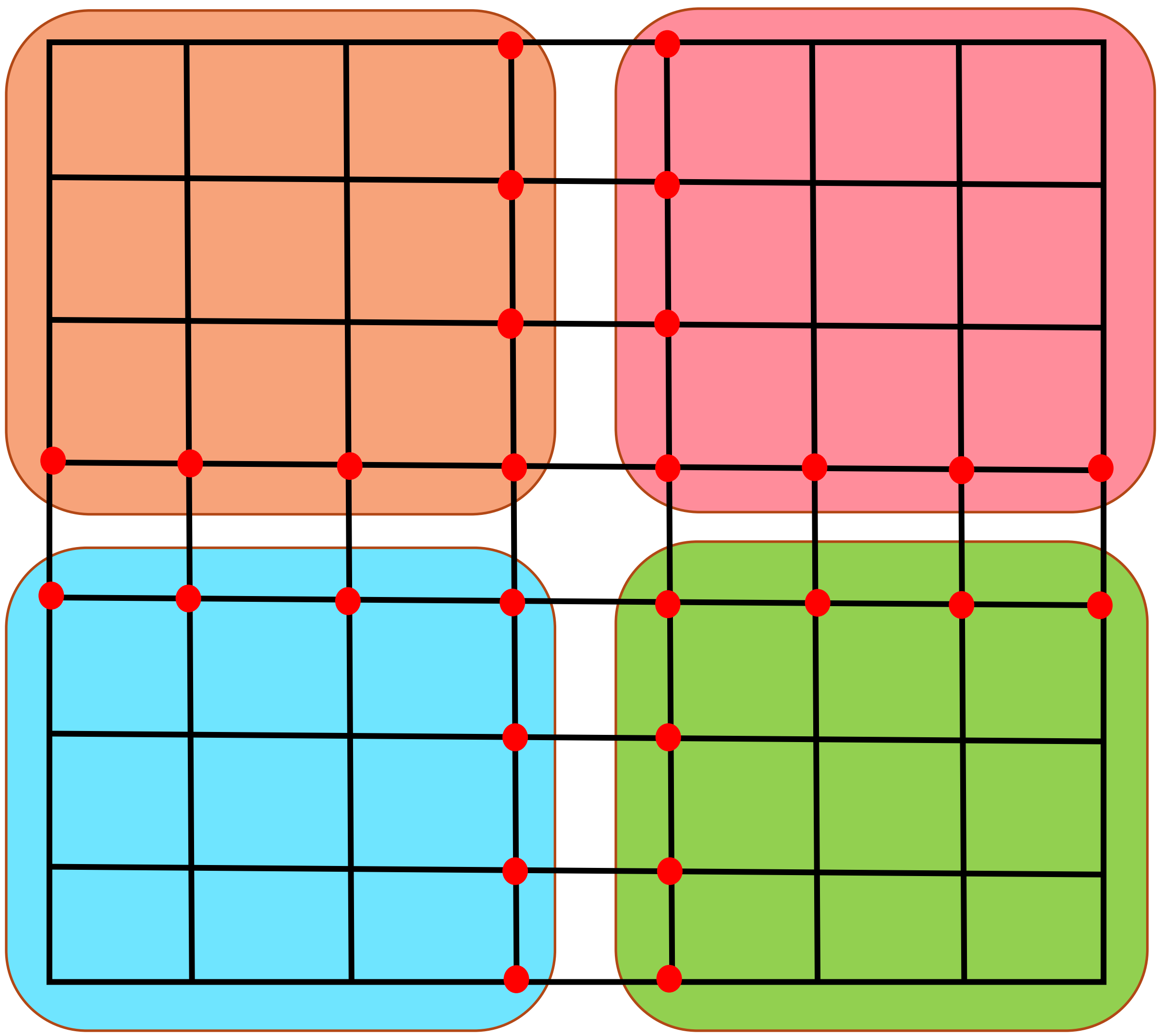
- The finite difference Laplacian operator is a sparse matrix operator on a vector of unknowns:

$$A\boldsymbol{q} = \boldsymbol{f}$$

- When the whole domain is distributed with MPI, each process will need to receive halo data to evaluate $A\boldsymbol{q}$
- Can compute Laplacian at interior stencil points while the halo data is being exchanged
- Jacobi Iterative method is the iteration:

$$\boldsymbol{q}^{k+1} = \boldsymbol{q}^{k} + D^{-1}\left(\boldsymbol{f} - A\boldsymbol{q}^{k}\right)$$

where D is the diagonal of A.



Jacobi Structure

- Initialize MPI
- Setup *Jacobi* object
  - ApplyTopology()
  - CreateMesh()
  - InitializeData()
- Execute *Jacobi* run method

```
//**************************************************************
//* Copyright (c) 2019, Advanced Micro Devices, Inc. All rights reserved.
//**************************************************************

#include "Jacobi.hpp"

int main(int argc, char ** argv)
{
  MPI_Init(&argc, &argv);

  MPI_Comm comm = MPI_COMM_WORLD;

  grid_t grid;
  mesh_t mesh;

  // Extract topology and domain dimensions from the command-line arguments
  ParseCommandLineArguments(argc, argv,
                            comm,
                            grid,
                            mesh);

  Jacobi_t Jacobi(grid, mesh);

  Jacobi.Run();

  // Finalize the MPI process
  MPI_Finalize();
  return STATUS_OK;
}
```

Jacobi Structure

- Initialize MPI
- Setup *Jacobi* object
  - ApplyTopology()
    - Establish MPI neighbors
    - Set GPU devices
  - CreateMesh()
  - InitializeData()
- Execute *Jacobi* run method

```
//select GPU
int devCount = 0;
SafeHipCall(hipGetDeviceCount(&devCount));

if (devCount == 0) {
  fprintf(stderr, "Error: MPI rank %d detected no GPUs."
                  "Terminating...\n", grid.rank);
  MPI_Abort(grid.comm, STATUS_ERR);
}

char name[255];
gethostname(name, 255);
long int hostId = gethostid();
long int* hostIds = (long int*) calloc(size,sizeof(long int));
MPI_Allgather(&hostId,1,MPI_LONG,hostIds,1,MPI_LONG,grid.comm);

//count how many ranks are on this node
int local_rank = 0;
for (int r=0;r<grid.rank;r++)
  if (hostIds[r]==hostId) local_rank++;

//select gpu via round-robin
grid.device_id = local_rank % devCount;
SafeHipCall(hipSetDevice(grid.device_id));
```


Jacobi Structure

- Initialize MPI
- Setup *Jacobi* object
  - ApplyTopology()
    - Establish MPI neighbors
    - Set GPU devices
  - CreateMesh()
    - Set local domain
    - Create halo exchange buffers
    - Create streams
  - InitializeData()
- Execute *Jacobi* run method

```
  //coordinates (including boundary points)
  mesh.x = (dfloat *) malloc((mesh.Nx+2)*sizeof(dfloat));
  mesh.y = (dfloat *) malloc((mesh.Ny+2)*sizeof(dfloat));

  for (int i=0;i<mesh.Nx;i++)
    mesh.x[i] = mesh.dx + grid.mycol*(DX+mesh.dx) + i*mesh.dx;
  for (int j=0;j<mesh.Ny;j++)
    mesh.y[j] = mesh.dy + grid.myrow*(DY+mesh.dy) + j*mesh.dy;

  //halo exchange storage
  SafeHipCall(hipHostMalloc(&(mesh.sendBuffer), mesh.Nhalo*sizeof(dfloat),0));
  SafeHipCall(hipHostMalloc(&(mesh.recvBuffer), mesh.Nhalo*sizeof(dfloat),0));
  SafeHipCall(hipMalloc(&(mesh.d_haloBuffer), mesh.Nhalo*sizeof(dfloat)));
  SafeHipCall(hipMemset(mesh.d_haloBuffer, 0, mesh.Nhalo*sizeof(dfloat)));

  for (int i=0;i<2*NSIDES;i++) mesh.requests[i] = MPI_REQUEST_NULL;

  //number of entries for each side in send/recv buffers
  mesh.sideLength[SIDE_DOWN ] = mesh.Nx;
  mesh.sideLength[SIDE_UP   ] = mesh.Nx;
  mesh.sideLength[SIDE_LEFT ] = mesh.Ny;
  mesh.sideLength[SIDE_RIGHT] = mesh.Ny;

  //offets of halo side data in send/recv buffers
  mesh.sideOffset[SIDE_DOWN ] = 0;
  mesh.sideOffset[SIDE_UP   ] = mesh.Nx;
  mesh.sideOffset[SIDE_LEFT ] = 2*mesh.Nx;
  mesh.sideOffset[SIDE_RIGHT] = 2*mesh.Nx + mesh.Ny;

  SafeHipCall(hipStreamCreate(&(computeStream)));
  SafeHipCall(hipStreamCreate(&(dataStream)));
}
```

Jacobi Structure

- Initialize MPI
- Setup *Jacobi* object
  - ApplyTopology()
    - Establish MPI neighbors
    - Set GPU devices
  - CreateMesh()
    - Set local domain
    - Create halo exchange buffers
    - Create streams
  - InitializeData()
    - Generate initial data
    - Copy data to device
- Execute *Jacobi* run method

```
//host buffers
h_U   = (dfloat*) malloc(mesh.N*sizeof(dfloat));
h_AU  = (dfloat*) malloc(mesh.N*sizeof(dfloat));
h_RHS = (dfloat*) malloc(mesh.N*sizeof(dfloat));
h_RES = (dfloat*) malloc(mesh.N*sizeof(dfloat));

for (int j=0;j<mesh.Ny;j++) {
  for (int i=0;i<mesh.Nx;i++) {
    int id = i+j*mesh.Nx;
    h_U[id] = 0.0; //initial guess
    h_RHS[id] = 0.0; //forcing
  }
}
```

⋮

```
// Setup boundary contributions
```

⋮

```
//device buffers
SafeHipCall(hipMalloc((void **)&d_U,   mesh.N*sizeof(dfloat)));
SafeHipCall(hipMalloc((void **)&d_AU,  mesh.N*sizeof(dfloat)));
SafeHipCall(hipMalloc((void **)&d_RHS, mesh.N*sizeof(dfloat)));
SafeHipCall(hipMalloc((void **)&d_RES, mesh.N*sizeof(dfloat)));

//send data to device
SafeHipCall(hipMemcpy(d_U, h_U, mesh.N*sizeof(dfloat), hipMemcpyHostToDevice));
SafeHipCall(hipMemcpy(d_AU, h_U, mesh.N*sizeof(dfloat), hipMemcpyHostToDevice));
SafeHipCall(hipMemcpy(d_RHS, h_RHS, mesh.N*sizeof(dfloat), hipMemcpyHostToDevice));
SafeHipCall(hipMemcpy(d_RES, h_RHS, mesh.N*sizeof(dfloat), hipMemcpyHostToDevice));

SafeHipCall(hipEventCreate(&JacobiLocalStart));
SafeHipCall(hipEventCreate(&JacobiLocalEnd));
```


Iterative Loop

```
while(){
    LocalLaplacian()
    HaloExchange()
    HaloLaplacian()
    JacobiIteration()
    Norm()
}
```

```
while ((iterations < JACOBI_MAX_LOOPS) && (residual > JACOBI_TOLERANCE)) {
  //queue local part of Laplacian to compute stream
  LocalLaplacian(grid, mesh, computeStream, d_U, d_AU);

  //Extract data off GPU exchange Halo with MPI
  HaloExchange(grid, mesh, dataStream, d_U);

  //use halo data to complete Laplacian computation
  HaloLaplacian(grid, mesh, computeStream, d_U, d_AU);

  //Jacobi iterative method
  // U = U + D^{-1}*(RHS - AU)
  JacobiIteration(grid, mesh, computeStream, d_RHS, d_AU, d_RES, d_U);

  //residual = ||U||
  residual = Norm(grid, mesh, computeStream, d_RES);

  //finish everything on device
  hipDeviceSynchronize();

  ++iterations;
}
```

- Note:
  - Two streams: computeStream and dataStream
  - hipEventRecord timings on each respective stream (omitted for clarity)

Preparing the code for the debugger

Preparing your HIP code for debugging:

- Use any optimization level you like, we'll use -O3
- Have ROCm load code objects at initialization:
  - `export HIP_ENABLE_DEFERRED_LOADING=0`
- Add **–ggdb** to your flags:
- 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-linux-gnu/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
```

Diving in

Launching the debugger, is the same as gdb:

```
rocgdb --args ./Jacobi_hip -g 1 1
```

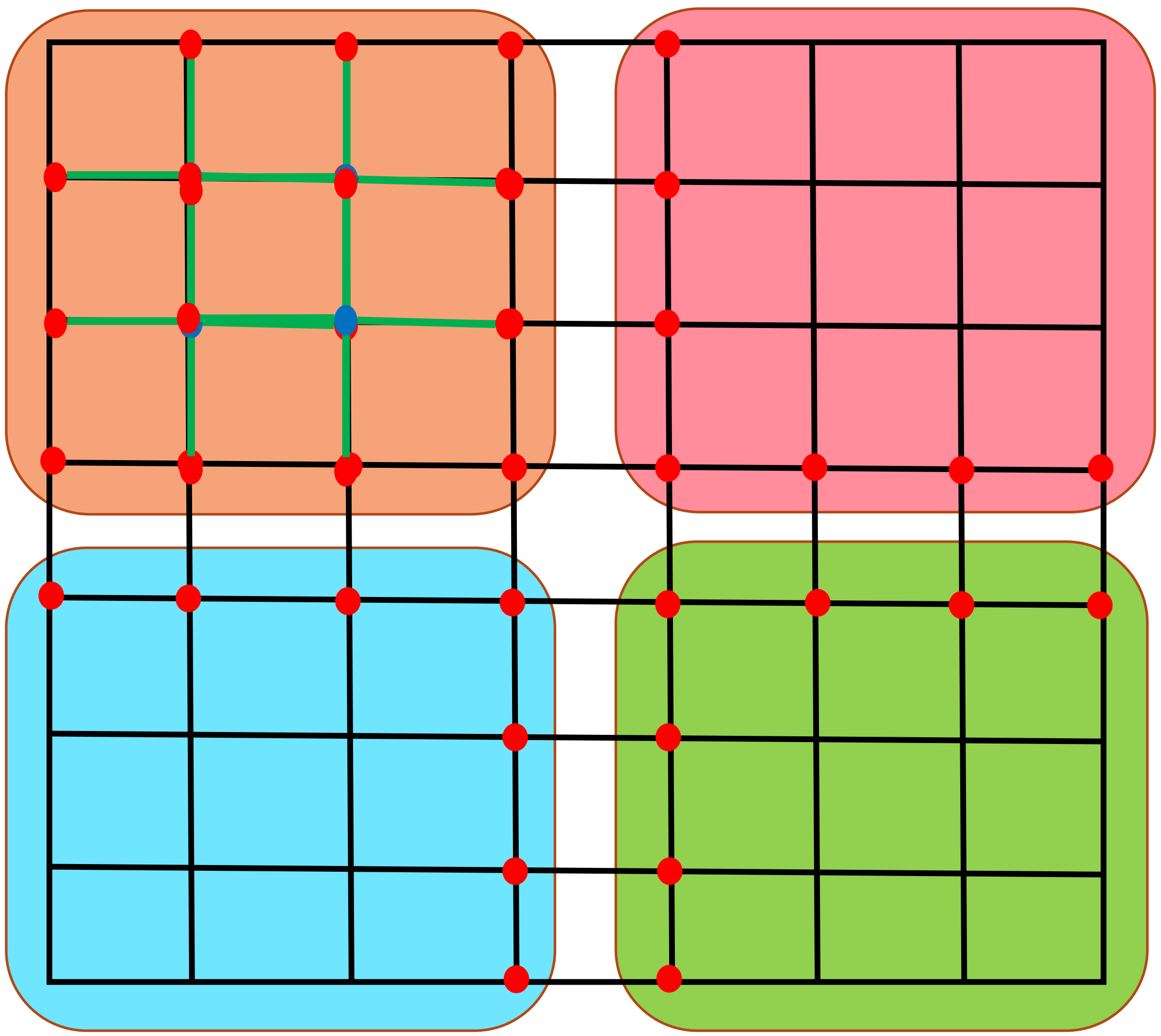
For this demo, I will be using a program called cgdb alongside rocgdb, this gives a nice curses-based interface, but is by no means required. In this case:

```
cgdb -d rocgdb --args ./Jacobi_hip -g 1 1
```

```
//**************************************************************************
//* Copyright (c) 2019, Advanced Micro Devices, Inc. All rights reserved.
//**************************************************************************

#include "Jacobi.hpp"

/**
 * @file JacobiMain.cpp
 * @brief This contains the application entry point
 */

/**
 * @brief The application entry point
 *
 * @param[in] argc  The number of command-line arguments
 * @param[in] argv  The list of command-line arguments
 */
int main(int argc, char ** argv)
{
  MPI_Init(&argc, &argv);

  MPI_Comm comm = MPI_COMM_WORLD;

  grid_t grid;
  mesh_t mesh;

  // Extract topology and domain dimensions from the command-line arguments
/home/nick/Documents/software_repos/hiptutorial/hip/JacobiMain.cpp
GNU gdb (rocm-rel-3.5-30) 9.1
Copyright (C) 2020 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 ./Jacobi_hip...
(gdb) s
```


Setting a breakpoint

Let's step into one of our kernels, `LocalLaplacianKernel` computes the finite difference for one of our MPI domains.



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-workstation: ~/Downloads/hiptutorial/hip

```
23
24     const int id_l = id - 1;
25     const int id_r = id + 1;
26     const int id_d = id - stride;
27     const int id_u = id + stride;
28
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);
31   }
32 }
33
34 void LocalLaplacian(grid_t& grid, mesh_t& 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
40-> int localNx = mesh.Nx-2;
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);
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./Jacobi_hip...
(gdb) b Laplacian.cpp: 40

Breakpoint 1 at 0x415080: file Laplacian.cpp, line 40.(gdb) run
Starting program: /home/jychang48/Downloads/hiptutorial/hip/Jacobi_hip -g 1 1
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Detaching after fork from child process 129946]
[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
[New Thread 0x7f28f7f53700 (LWP 129955)]
Rank 0 selecting device 0 on host jychang48-workstation
[New Thread 0x7f28f71ff700 (LWP 129956)]
[Thread 0x7f28f71ff700 (LWP 129956) exited]
[New Thread 0x7f28fc07f700 (LWP 129957)]
[New Thread 0x7f28f7752700 (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=..., stream=0
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)
```


Setting a breakpoint in host code

Typing ‘step’ enables one to iterate through the stack trace:

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
33
34  void LocalLaplacian(grid_t& grid, mesh_t& 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
40    int localNx = mesh.Nx-2;
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);
57  }
58
59
60  // AU_i,j = (-U_i+1,j + 2U_i,j - U_i-1,j)/dx^2 +
61  //          (-U_i,j+1 + 2U_i,j - U_i,j-1)/dy^2
62  __global__ void HaloLaplacianKernel(const int Nx,
63                                      const int Ny,
64                                      const int stride,
65                                      const dfloat dx,
66                                      const dfloat dy,
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./Jacobi_hip...
(gdb) b Laplacian.cpp: 40

Breakpoint 1 at 0x415080: file Laplacian.cpp, line 40.(gdb) run
Starting program: /home/jychang48/Downloads/hiptutorial/hip/Jacobi_hip -g 1 1
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Detaching after fork from child process 129946]
[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
[New Thread 0x7f28f7f53700 (LWP 129955)]
Rank 0 selecting device 0 on host jychang48-workstation
[New Thread 0x7f28f71ff700 (LWP 129956)]
[Thread 0x7f28f71ff700 (LWP 129956) exited]
[New Thread 0x7f28fc07f700 (LWP 129957)]
[New Thread 0x7f28f7752700 (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=..., stream=0
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)
```


Setting a breakpoint in device kernel

Invoke ‘b’ or ‘break’ to the device kernel of interest:

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
 1  //*****************************************************************************
 2  //* Copyright (c) 2019, Advanced Micro Devices, Inc. All rights reserved.
 3  //*****************************************************************************
 4
 5  #include "Jacobi.hpp"
 6
 7  // AU_i,j = (-U_i+1,j + 2U_i,j - U_i-1,j)/dx^2 +
 8  //          (-U_i,j+1 + 2U_i,j - U_i,j-1)/dy^2
 9 >__global__ void LocalLaplacianKernel(const int localNx,
10                                    const int localNy,
11                                    const int stride,
12                                    const dfloat dx,
13                                    const dfloat dy,
14                                    const dfloat *__restrict__ U,
15                                          dfloat *__restrict__ AU) {
16
17    const int i = threadIdx.x+blockIdx.x*blockDim.x;
18    const int j = threadIdx.y+blockIdx.y*blockDim.y;
19
20    if ((i<localNx) && (j<localNy)) {
21
22      const int id = (i+1) + (j+1)*stride;
23
24      const int id_l = id - 1;
25      const int id_r = id + 1;
26      const int id_d = id - stride;
27      const int id_u = id + stride;
28
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);
31    }
32  }
33
34  void LocalLaplacian(grid_t& grid, mesh_t& mesh,
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
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 0x7f28f71ff700 (LWP 129956)]
[Thread 0x7f28f71ff700 (LWP 129956) exited]
[New Thread 0x7f28fc07f700 (LWP 129957)]
[New Thread 0x7f28f7752700 (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=..., stream=0
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.
Make breakpoint pending on future shared library load? (y or [n]) y

Breakpoint 2 (LocalLaplacianKernel) pending.(gdb) continue
Continuing.
[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
(gdb)
```

Setting a breakpoint in device kernel

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

:

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```cpp
//**************************************************************************
//* Copyright (c) 2019, Advanced Micro Devices, Inc. All rights reserved.
//**************************************************************************

#include "Jacobi.hpp"

// AU_i,j = (-U_i+1,j + 2U_i,j - U_i-1,j)/dx^2 +
//          (-U_i,j+1 + 2U_i,j - U_i,j-1)/dy^2
>__global__ void LocalLaplacianKernel(const int localNx,
                                     const int localNy,
                                     const int stride,
                                     const dfloat dx,
                                     const dfloat dy,
                                     const dfloat *__restrict__ U,
                                           dfloat *__restrict__ AU) {

  const int i = threadIdx.x+blockIdx.x*blockDim.x;
  const int j = threadIdx.y+blockIdx.y*blockDim.y;

  if ((i<localNx) && (j<localNy)) {

    const int id = (i+1) + (j+1)*stride;

    const int id_l = id - 1;
    const int id_r = id + 1;
    const int id_d = id - stride;
    const int id_u = id + stride;

    AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +
             (-U[id_d] + 2*U[id] - U[id_u])/(dy*dy);
  }
}

void LocalLaplacian(grid_t& grid, mesh_t& mesh,
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
$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.
Make breakpoint pending on future shared library load? (y or [n]) y

Breakpoint 2 (LocalLaplacianKernel) pending.(gdb) continue
Continuing.
[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
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1287 (65,1,0)/2]

Thread 1295 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1269 (61,1,0)/0]

Thread 1277 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:363 (90,0,0)/2]

Thread 371 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized o
ut>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optimi
zed out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb)
```

Setting a breakpoint in device kernel

```
[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
```

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


Setting a breakpoint in device kernel

- `agent-id` = Agent Target ID

```
[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
```

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


Setting a breakpoint in device kernel

- `agent-id` = Agent Target ID
- `queue-id` = Queue Target ID

```
[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
```

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

Setting a breakpoint in device kernel

- `agent-id` = Agent Target ID
- `queue-id` = Queue Target ID
- `dispatch-num` = Dispatch Target ID – how many kernels have been launched

```
[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
```

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

Setting a breakpoint in device kernel

- `agent-id` = Agent Target ID
- `queue-id` = Queue Target ID
- `dispatch-num` = Dispatch Target ID – how many kernels have been launched
- `wave-id` = Wavefront ID – index of wavefront of kernel

```
[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
```

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

Setting a breakpoint in device kernel

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

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

Setting a breakpoint in device kernel

To look at the progress of a single wavefront, disable the breakpoint by typing ‘disable *<num>*’, and then ‘step’

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
//***********************************************************************
//* Copyright (c) 2019, Advanced Micro Devices, Inc. All rights reserved.
//***********************************************************************

#include "Jacobi.hpp"

// AU_i,j = (-U_i+1,j + 2U_i,j - U_i-1,j)/dx^2 +
//          (-U_i,j+1 + 2U_i,j - U_i,j-1)/dy^2
__global__ void LocalLaplacianKernel(const int localNx,
                                     const int localNy,
                                     const int stride,
                                     const dfloat dx,
                                     const dfloat dy,
                                     const dfloat *__restrict__ U,
                                           dfloat *__restrict__ AU) {

-> const int i = threadIdx.x+blockIdx.x*blockDim.x;
   const int j = threadIdx.y+blockIdx.y*blockDim.y;

   if ((i<localNx) && (j<localNy)) {

     const int id = (i+1) + (j+1)*stride;

     const int id_l = id - 1;
     const int id_r = id + 1;
     const int id_d = id - stride;
     const int id_u = id + stride;

     AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +
              (-U[id_d] + 2*U[id] - U[id_u])/(dy*dy);
   }
 }

 void LocalLaplacian(grid_t& grid, mesh_t& mesh,
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
$3 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
(gdb) step
(gdb) step
(gdb) b LocalLaplacianKernel
Function "LocalLaplacianKernel" not defined.
Make breakpoint pending on future shared library load? (y or [n]) y

Breakpoint 2 (LocalLaplacianKernel) pending.(gdb) continue
Continuing.
[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
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1287 (65,1,0)/2]

Thread 1295 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1269 (61,1,0)/0]

Thread 1277 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:363 (90,0,0)/2]

Thread 371 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized o
ut>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optimi
zed out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) disable 2
(gdb) step
(gdb)
```


Setting a breakpoint in device kernel

Use ‘stepi’ which enables the user to dive deeper into the HIP API

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
215       } while (0)
216
217   #define hipLaunchKernelGGL(kernelName, ...)  hipLaunchKernelGGLInternal((kernelN
218   #endif
219
220   #include <hip/hip_runtime_api.h>
221   extern "C" __device__ __attribute__((const)) size_t __ockl_get_local_id(uint);
222   extern "C" __device__ __attribute__((const)) size_t __ockl_get_group_id(uint);
223   extern "C" __device__ __attribute__((const)) size_t __ockl_get_local_size(uint);
224   extern "C" __device__ __attribute__((const)) size_t __ockl_get_num_groups(uint);
225   struct __HIP_BlockIdx {
226     __device__
227     std::uint32_t operator()(std::uint32_t x) const noexcept { return __ockl_get_g
228   };
229   struct __HIP_BlockDim {
230     __device__
231     std::uint32_t operator()(std::uint32_t x) const noexcept {
232 --->  return __ockl_get_local_size(x);
233     }
234   };
235   struct __HIP_GridDim {
236     __device__
237     std::uint32_t operator()(std::uint32_t x) const noexcept {
238       return __ockl_get_num_groups(x);
239     }
240   };
241   struct __HIP_ThreadIdx {
242     __device__
243     std::uint32_t operator()(std::uint32_t x) const noexcept {
244       return __ockl_get_local_id(x);
245     }
246   };
247
248   template <typename F>
/opt/rocm-4.2.0/hip/include/hip/amd_detail/hip_runtime.h
```

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

Breakpoint 2 (LocalLaplacianKernel) pending.(gdb) continue
Continuing.
[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
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1287 (65,1,0)/2]

Thread 1295 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1269 (61,1,0)/0]

Thread 1277 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:363 (90,0,0)/2]

Thread 371 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized o
ut>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optimi
zed out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) disable 2
(gdb) step
(gdb) stepi
0x00007f28f722f010 in __HIP_BlockDim::operator() (this=<optimized out>, x=<optimized
 out>) at /opt/rocm-4.2.0/hip/include/hip/amd_detail/hip_runtime.h:232
(gdb)
```


Setting a breakpoint in device kernel

Type ‘step’ again and you can now freely step through the device kernel

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
 1  //***************************************************************************
 2  //* Copyright (c) 2019, Advanced Micro Devices, Inc. All rights reserved.
 3  //***************************************************************************
 4
 5  #include "Jacobi.hpp"
 6
 7  // AU_i,j = (-U_i+1,j + 2U_i,j - U_i-1,j)/dx^2 +
 8  //          (-U_i,j+1 + 2U_i,j - U_i,j-1)/dy^2
 9  __global__ void LocalLaplacianKernel(const int localNx,
10                                       const int localNy,
11                                       const int stride,
12                                       const dfloat dx,
13                                       const dfloat dy,
14                                       const dfloat *__restrict__ U,
15                                             dfloat *__restrict__ AU) {
16
17    const int i = threadIdx.x+blockIdx.x*blockDim.x;
18 -> const int j = threadIdx.y+blockIdx.y*blockDim.y;
19
20    if ((i<localNx) && (j<localNy)) {
21
22      const int id = (i+1) + (j+1)*stride;
23
24      const int id_l = id - 1;
25      const int id_r = id + 1;
26      const int id_d = id - stride;
27      const int id_u = id + stride;
28
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);
31    }
32  }
33
34  void LocalLaplacian(grid_t& grid, mesh_t& mesh,
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
Make breakpoint pending on future shared library load? (y or [n]) y

Breakpoint 2 (LocalLaplacianKernel) pending.(gdb) continue
Continuing.
[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
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1287 (65,1,0)/2]

Thread 1295 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1269 (61,1,0)/0]

Thread 1277 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:363 (90,0,0)/2]

Thread 371 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized o
ut>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optimi
zed out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) disable 2
(gdb) step
(gdb) stepi
0x00007f28f722f010 in __HIP_BlockDim::operator() (this=<optimized out>, x=<optimized
 out>) at /opt/rocm-4.2.0/hip/include/hip/amd_detail/hip_runtime.h:232
(gdb) step
(gdb) step
(gdb)
```


AMD

Examining the ISA

Several ways one can view the ISA. Using cgdb, type

ESC -> :set dis -> ENTER

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
9         __global__ void LocalLaplacianKernel(const int localNx,
   0x00007f28f722f000 <+0>:     s_load_dwordx4 s[0:3], s[6:7], 0x0

/opt/rocm-4.2.0/hip/include/hip/amd_detail/hip_runtime.h:
232         return __ockl_get_local_size(x);
   0x00007f28f722f008 <+8>:     s_load_dword s10, s[4:5], 0x4
   0x00007f28f722f010 <+16>:    s_waitcnt lgkmcnt(0)
   0x00007f28f722f014 <+20>:    s_lshr_b32 s3, s10, 16
   0x00007f28f722f018 <+24>:    s_and_b32 s4, s10, 0xffff
   0x00007f28f722f020 <+32>:    s_mul_i32 s8, s8, s4
   0x00007f28f722f024 <+36>:    s_mul_i32 s9, s9, s3

Laplacian.cpp:
17        const int i = threadIdx.x+blockIdx.x*blockDim.x;
   0x00007f28f722f028 <+40>:    v_add_u32_e32 v0, s8, v0

18        const int j = threadIdx.y+blockIdx.y*blockDim.y;
──> 0x00007f28f722f02c <+44>:   v_add_u32_e32 v1, s9, v1

19
20        if ((i<localNx) && (j<localNy)) {
   0x00007f28f722f030 <+48>:    v_cmp_gt_i32_e32 vcc, s0, v0
   0x00007f28f722f034 <+52>:    v_cmp_gt_i32_e64 s[0:1], s1, v1
   0x00007f28f722f03c <+60>:    s_and_b64 s[0:1], vcc, s[0:1]
   0x00007f28f722f040 <+64>:    s_and_saveexec_b64 s[4:5], s[0:1]
   0x00007f28f722f044 <+68>:    s_cbranch_execz 104

21
22          const int id = (i+1) + (j+1)*stride;
   0x00007f28f722f048 <+72>:    v_add_u32_e32 v1, 1, v1
   0x00007f28f722f04c <+76>:    v_mul_lo_u32 v1, v1, s2
   0x00007f28f722f054 <+84>:    s_load_dwordx2 s[0:1], s[6:7], 0x20

9         __global__ void LocalLaplacianKernel(const int localNx,
>de for function _Z20LocalLaplacianKerneliiiddPKdPd: (7f28f722f000 - 7f28f722f1e8) **
```

```
Make breakpoint pending on future shared library load? (y or [n]) y

Breakpoint 2 (LocalLaplacianKernel) pending.(gdb) continue
Continuing.
[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
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1287 (65,1,0)/2]

Thread 1295 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1269 (61,1,0)/0]

Thread 1277 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:363 (90,0,0)/2]

Thread 371 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized o
ut>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optimi
zed out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) disable 2
(gdb) step
(gdb) stepi
0x00007f28f722f010 in __HIP_BlockDim::operator() (this=<optimized out>, x=<optimized
 out>) at /opt/rocm-4.2.0/hip/include/hip/amd_detail/hip_runtime.h:232
(gdb) step
(gdb) step
(gdb)
```


AMD

Examining the ISA

Let's say I want to see what the value of "i" is for my wavefront:

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
9          __global__ void LocalLaplacianKernel(const int localNx,
   0x00007f28f722f000 <+0>:     s_load_dwordx4 s[0:3], s[6:7], 0x0

/opt/rocm-4.2.0/hip/include/hip/amd_detail/hip_runtime.h:
232         return __ockl_get_local_size(x);
   0x00007f28f722f008 <+8>:     s_load_dword s10, s[4:5], 0x4
   0x00007f28f722f010 <+16>:    s_waitcnt lgkmcnt(0)
   0x00007f28f722f014 <+20>:    s_lshr_b32 s3, s10, 16
   0x00007f28f722f018 <+24>:    s_and_b32 s4, s10, 0xffff
   0x00007f28f722f020 <+32>:    s_mul_i32 s8, s8, s4
   0x00007f28f722f024 <+36>:    s_mul_i32 s9, s9, s3

Laplacian.cpp:
17         const int i = threadIdx.x+blockIdx.x*blockDim.x;
   0x00007f28f722f028 <+40>:    v_add_u32_e32 v0, s8, v0

18         const int j = threadIdx.y+blockIdx.y*blockDim.y;
──> 0x00007f28f722f02c <+44>:    v_add_u32_e32 v1, s9, v1

19
20         if ((i<localNx) && (j<localNy)) {
   0x00007f28f722f030 <+48>:    v_cmp_gt_i32_e32 vcc, s0, v0
   0x00007f28f722f034 <+52>:    v_cmp_gt_i32_e64 s[0:1], s1, v1
   0x00007f28f722f03c <+60>:    s_and_b64 s[0:1], vcc, s[0:1]
   0x00007f28f722f040 <+64>:    s_and_saveexec_b64 s[4:5], s[0:1]
   0x00007f28f722f044 <+68>:    s_cbranch_execz 104

21
22           const int id = (i+1) + (j+1)*stride;
   0x00007f28f722f048 <+72>:    v_add_u32_e32 v1, 1, v1
   0x00007f28f722f04c <+76>:    v_mul_lo_u32 v1, v1, s2
   0x00007f28f722f054 <+84>:    s_load_dwordx2 s[0:1], s[6:7], 0x20

9          __global__ void LocalLaplacianKernel(const int localNx,
>de for function _Z20LocalLaplacianKerneliiiddPKdPd: (7f28f722f000 - 7f28f722f1e8) **
```

```
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1287 (65,1,0)/2]

Thread 1295 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:1269 (61,1,0)/0]

Thread 1277 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:363 (90,0,0)/2]

Thread 371 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized o
ut>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optimi
zed out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) disable 2
(gdb) step
(gdb) stepi
0x00007f28f722f010 in __HIP_BlockDim::operator() (this=<optimized out>, x=<optimized
 out>) at /opt/rocm-4.2.0/hip/include/hip/amd_detail/hip_runtime.h:232
(gdb) step
(gdb) step
(gdb) info v0
Undefined info command: "v0".  Try "help info".
(gdb) info reg v0
v0             {0x5a0, 0x5a1, 0x5a2, 0x5a3, 0x5a4, 0x5a5, 0x5a6, 0x5a7, 0x5a8, 0x5a9
, 0x5aa, 0x5ab, 0x5ac, 0x5ad, 0x5ae, 0x5af, 0x5a0, 0x5a1, 0x5a2, 0x5a3, 0x5a4, 0x5a5
, 0x5a6, 0x5a7, 0x5a8, 0x5a9, 0x5aa, 0x5ab, 0x5ac, 0x5ad, 0x5ae, 0x5af, 0x5a0, 0x5a1
, 0x5a2, 0x5a3, 0x5a4, 0x5a5, 0x5a6, 0x5a7, 0x5a8, 0x5a9, 0x5aa, 0x5ab, 0x5ac, 0x5ad
, 0x5ae, 0x5af, 0x5a0, 0x5a1, 0x5a2, 0x5a3, 0x5a4, 0x5a5, 0x5a6, 0x5a7, 0x5a8, 0x5a9
, 0x5aa, 0x5ab, 0x5ac, 0x5ad, 0x5ae, 0x5af}
(gdb)
```


Examining the ISA

This also works for scalar registers, for instance here we can check the conditional:

```
(i < localNx) && (j < localNy)
```

which is stored in s[0:1], one bit corresponding to each thread in the wavefront:

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
Laplacian.cpp:
17          const int i = threadIdx.x+blockIdx.x*blockDim.x;
   0x00007f28f722f028 <+40>:    v_add_u32_e32 v0, s8, v0

18          const int j = threadIdx.y+blockIdx.y*blockDim.y;
   0x00007f28f722f02c <+44>:    v_add_u32_e32 v1, s9, v1

19
20          if ((i<localNx) && (j<localNy)) {
   0x00007f28f722f030 <+48>:    v_cmp_gt_i32_e32 vcc, s0, v0
   0x00007f28f722f034 <+52>:    v_cmp_gt_i32_e64 s[0:1], s1, v1
   0x00007f28f722f03c <+60>:    s_and_b64 s[0:1], vcc, s[0:1]
   0x00007f28f722f040 <+64>:    s_and_saveexec_b64 s[4:5], s[0:1]
   0x00007f28f722f044 <+68>:    s_cbranch_execz 104

21
22            const int id = (i+1) + (j+1)*stride;
-->0x00007f28f722f048 <+72>:    v_add_u32_e32 v1, 1, v1
   0x00007f28f722f04c <+76>:    v_mul_lo_u32 v1, v1, s2
   0x00007f28f722f054 <+84>:    s_load_dwordx2 s[0:1], s[6:7], 0x20

9         __global__ void LocalLaplacianKernel(const int localNx,
   0x00007f28f722f05c <+92>:    s_load_dwordx4 s[8:11], s[6:7], 0x10

21
22            const int id = (i+1) + (j+1)*stride;
   0x00007f28f722f064 <+100>:   v_add_u32_e32 v4, v1, v0

28
29            AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +
   0x00007f28f722f068 <+104>:   v_ashrrev_i32_e32 v5, 31, v4
   0x00007f28f722f06c <+108>:   v_lshlrev_b64 v[5:6], 3, v[4:5]
   0x00007f28f722f074 <+116>:   s_waitcnt lgkmcnt(0)
   0x00007f28f722f078 <+120>:   v_mov_b32_e32 v15, s1
>de for function _Z20LocalLaplacianKerneliiiddPKdPd: (7f28f722f000 - 7f28f722f1e8) **
```

```
out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optim
ized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) step
[Switching to AMDGPU Thread 1:5:1:363 (90,0,0)/2]

Thread 371 "Jacobi_hip" hit Breakpoint 2, LocalLaplacianKernel (localNx=<optimized o
ut>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out>, dy=<optimi
zed out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
(gdb) disable 2
(gdb) step
(gdb) stepi
0x00007f28f722f010 in __HIP_BlockDim::operator() (this=<optimized out>, x=<optimized
 out>) at /opt/rocm-4.2.0/hip/include/hip/amd_detail/hip_runtime.h:232
(gdb) step
(gdb) step
(gdb) info v0
Undefined info command: "v0".  Try "help info".
(gdb) info reg v0
v0             {0x5a0, 0x5a1, 0x5a2, 0x5a3, 0x5a4, 0x5a5, 0x5a6, 0x5a7, 0x5a8, 0x5a9
, 0x5aa, 0x5ab, 0x5ac, 0x5ad, 0x5ae, 0x5af, 0x5a0, 0x5a1, 0x5a2, 0x5a3, 0x5a4, 0x5a5
, 0x5a6, 0x5a7, 0x5a8, 0x5a9, 0x5aa, 0x5ab, 0x5ac, 0x5ad, 0x5ae, 0x5af, 0x5a0, 0x5a1
, 0x5a2, 0x5a3, 0x5a4, 0x5a5, 0x5a6, 0x5a7, 0x5a8, 0x5a9, 0x5aa, 0x5ab, 0x5ac, 0x5ad
, 0x5ae, 0x5af, 0x5a0, 0x5a1, 0x5a2, 0x5a3, 0x5a4, 0x5a5, 0x5a6, 0x5a7, 0x5a8, 0x5a9
, 0x5aa, 0x5ab, 0x5ac, 0x5ad, 0x5ae, 0x5af}
(gdb) info reg s0
s0             0xfe                4094
(gdb) info reg s1
s1             0xfe                4094
(gdb) step
(gdb) step
(gdb) info reg s0
s0             0xffffffff          -1
(gdb) info reg s1
s1             0xffffffff          -1
(gdb)
```


AMD

Switching wavefronts

Now that I've been stepping only my wavefront, are the others still at the beginning of the kernel? Use info threads. This enables us t o see the location of both host threads and GPU wavefronts …

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
Laplacian.cpp:
17          const int i = threadIdx.x+blockIdx.x*blockDim.x;
   0x00007f28f722f028 <+40>:    v_add_u32_e32 v0, s8, v0

18          const int j = threadIdx.y+blockIdx.y*blockDim.y;
   0x00007f28f722f02c <+44>:    v_add_u32_e32 v1, s9, v1

19
20          if ((i<localNx) && (j<localNy)) {
   0x00007f28f722f030 <+48>:    v_cmp_gt_i32_e32 vcc, s0, v0
   0x00007f28f722f034 <+52>:    v_cmp_gt_i32_e64 s[0:1], s1, v1
   0x00007f28f722f03c <+60>:    s_and_b64 s[0:1], vcc, s[0:1]
   0x00007f28f722f040 <+64>:    s_and_saveexec_b64 s[4:5], s[0:1]
   0x00007f28f722f044 <+68>:    s_cbranch_execz 104

21
22            const int id = (i+1) + (j+1)*stride;
-->0x00007f28f722f048 <+72>:    v_add_u32_e32 v1, 1, v1
   0x00007f28f722f04c <+76>:    v_mul_lo_u32 v1, v1, s2
   0x00007f28f722f054 <+84>:    s_load_dwordx2 s[0:1], s[6:7], 0x20

9         __global__ void LocalLaplacianKernel(const int localNx,
   0x00007f28f722f05c <+92>:    s_load_dwordx4 s[8:11], s[6:7], 0x10

21
22            const int id = (i+1) + (j+1)*stride;
   0x00007f28f722f064 <+100>:   v_add_u32_e32 v4, v1, v0

28
29            AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +
   0x00007f28f722f068 <+104>:   v_ashrrev_i32_e32 v5, 31, v4
   0x00007f28f722f06c <+108>:   v_lshlrev_b64 v[5:6], 3, v[4:5]
   0x00007f28f722f074 <+116>:   s_waitcnt lgkmcnt(0)
   0x00007f28f722f078 <+120>:   v_mov_b32_e32 v15, s1
>de for function _Z20LocalLaplacianKerneliiiddPKdPd: (7f28f722f000 - 7f28f722f1e8) **
```

```
(gdb) info threads                                                          [110/112]
  Id   Target Id                                          Frame
  1    Thread 0x7f291814d8c0 (LWP 129937) "Jacobi_hip" 0x00007f2918b7a89b in sched_y
ield () from /lib/x86_64-linux-gnu/libc.so.6
  2    Thread 0x7f2900108700 (LWP 129953) "Jacobi_hip" 0x00007f2918b8aaff in poll ()
 from /lib/x86_64-linux-gnu/libc.so.6
  3    Thread 0x7f28ff74a700 (LWP 129954) "Jacobi_hip" 0x00007f2918b975ce in epoll_w
ait () from /lib/x86_64-linux-gnu/libc.so.6
  4    Thread 0x7f28f7f53700 (LWP 129955) "Jacobi_hip" 0x00007f2918b8c50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  6    Thread 0x7f28fc07f700 (LWP 129957) "Jacobi_hip" 0x00007f2918b7a89b in sched_y
ield () from /lib/x86_64-linux-gnu/libc.so.6
  7    Thread 0x7f28f7752700 (LWP 129958) "Jacobi_hip" 0x00007f2918b8c50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  8    Thread 0x7f28f757f700 (LWP 129959) "Jacobi_hip" 0x00007f291a2f4678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
* 371  AMDGPU Thread 1:5:1:363 (90,0,0)/2 "Jacobi_hip" LocalLaplacianKernel (localNx
=<optimized out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out
>, dy=<optimized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:22
(gdb)
  Id   Target Id                                          Frame
  1    Thread 0x7f291814d8c0 (LWP 129937) "Jacobi_hip" 0x00007f2918b7a89b in sched_y
ield () from /lib/x86_64-linux-gnu/libc.so.6
  2    Thread 0x7f2900108700 (LWP 129953) "Jacobi_hip" 0x00007f2918b8aaff in poll ()
 from /lib/x86_64-linux-gnu/libc.so.6
  3    Thread 0x7f28ff74a700 (LWP 129954) "Jacobi_hip" 0x00007f2918b975ce in epoll_w
ait () from /lib/x86_64-linux-gnu/libc.so.6
  4    Thread 0x7f28f7f53700 (LWP 129955) "Jacobi_hip" 0x00007f2918b8c50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  6    Thread 0x7f28fc07f700 (LWP 129957) "Jacobi_hip" 0x00007f2918b7a89b in sched_y
ield () from /lib/x86_64-linux-gnu/libc.so.6
  7    Thread 0x7f28f7752700 (LWP 129958) "Jacobi_hip" 0x00007f2918b8c50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  8    Thread 0x7f28f757f700 (LWP 129959) "Jacobi_hip" 0x00007f291a2f4678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
```

Switching wavefronts

.. but where did all the other threads go? **Step allows other wavefronts to advance**, hence the rest of the waves in our kernel have already completed!:

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
Laplacian.cpp:
17          const int i = threadIdx.x+blockIdx.x*blockDim.x;
   0x00007f28f722f028 <+40>:    v_add_u32_e32 v0, s8, v0

18          const int j = threadIdx.y+blockIdx.y*blockDim.y;
   0x00007f28f722f02c <+44>:    v_add_u32_e32 v1, s9, v1

19
20          if ((i<localNx) && (j<localNy)) {
   0x00007f28f722f030 <+48>:    v_cmp_gt_i32_e32 vcc, s0, v0
   0x00007f28f722f034 <+52>:    v_cmp_gt_i32_e64 s[0:1], s1, v1
   0x00007f28f722f03c <+60>:    s_and_b64 s[0:1], vcc, s[0:1]
   0x00007f28f722f040 <+64>:    s_and_saveexec_b64 s[4:5], s[0:1]
   0x00007f28f722f044 <+68>:    s_cbranch_execz 104

21
22            const int id = (i+1) + (j+1)*stride;
 --> 0x00007f28f722f048 <+72>:    v_add_u32_e32 v1, 1, v1
   0x00007f28f722f04c <+76>:    v_mul_lo_u32 v1, v1, s2
   0x00007f28f722f054 <+84>:    s_load_dwordx2 s[0:1], s[6:7], 0x20

9         __global__ void LocalLaplacianKernel(const int localNx,
   0x00007f28f722f05c <+92>:    s_load_dwordx4 s[8:11], s[6:7], 0x10

21
22            const int id = (i+1) + (j+1)*stride;
   0x00007f28f722f064 <+100>:   v_add_u32_e32 v4, v1, v0

28
29            AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +
   0x00007f28f722f068 <+104>:   v_ashrrev_i32_e32 v5, 31, v4
   0x00007f28f722f06c <+108>:   v_lshlrev_b64 v[5:6], 3, v[4:5]
   0x00007f28f722f074 <+116>:   s_waitcnt lgkmcnt(0)
   0x00007f28f722f078 <+120>:   v_mov_b32_e32 v15, s1
>de for function _Z20LocalLaplacianKerneliiiddPKdPd: (7f28f722f000 - 7f28f722f1e8) **
```

```
(gdb) info threads                                                              [110/112]
  Id   Target Id                                                Frame
  1    Thread 0x7f291814d8c0 (LWP 129937) "Jacobi_hip" 0x00007f2918b7a89b in sched_y
ield () from /lib/x86_64-linux-gnu/libc.so.6
  2    Thread 0x7f2900108700 (LWP 129953) "Jacobi_hip" 0x00007f2918b8aaff in poll ()
 from /lib/x86_64-linux-gnu/libc.so.6
  3    Thread 0x7f28ff74a700 (LWP 129954) "Jacobi_hip" 0x00007f2918b975ce in epoll_w
ait () from /lib/x86_64-linux-gnu/libc.so.6
  4    Thread 0x7f28f7f53700 (LWP 129955) "Jacobi_hip" 0x00007f2918b8c50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  6    Thread 0x7f28fc07f700 (LWP 129957) "Jacobi_hip" 0x00007f2918b7a89b in sched_y
ield () from /lib/x86_64-linux-gnu/libc.so.6
  7    Thread 0x7f28f7752700 (LWP 129958) "Jacobi_hip" 0x00007f2918b8c50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  8    Thread 0x7f28f757f700 (LWP 129959) "Jacobi_hip" 0x00007f291a2f4678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
* 371  AMDGPU Thread 1:5:1:363 (90,0,0)/2 "Jacobi_hip" LocalLaplacianKernel (localNx
=<optimized out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized out
>, dy=<optimized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:22
(gdb)
  Id   Target Id                                                Frame
  1    Thread 0x7f291814d8c0 (LWP 129937) "Jacobi_hip" 0x00007f2918b7a89b in sched_y
ield () from /lib/x86_64-linux-gnu/libc.so.6
  2    Thread 0x7f2900108700 (LWP 129953) "Jacobi_hip" 0x00007f2918b8aaff in poll ()
 from /lib/x86_64-linux-gnu/libc.so.6
  3    Thread 0x7f28ff74a700 (LWP 129954) "Jacobi_hip" 0x00007f2918b975ce in epoll_w
ait () from /lib/x86_64-linux-gnu/libc.so.6
  4    Thread 0x7f28f7f53700 (LWP 129955) "Jacobi_hip" 0x00007f2918b8c50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  6    Thread 0x7f28fc07f700 (LWP 129957) "Jacobi_hip" 0x00007f2918b7a89b in sched_y
ield () from /lib/x86_64-linux-gnu/libc.so.6
  7    Thread 0x7f28f7752700 (LWP 129958) "Jacobi_hip" 0x00007f2918b8c50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  8    Thread 0x7f28f757f700 (LWP 129959) "Jacobi_hip" 0x00007f291a2f4678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
```


AMD

Rerunning rocgdb with scheduler-locking

Use ‘set scheduler-locking on’ will not continue the other waves.

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
//**************************************************************************
//* Copyright (c) 2019, Advanced Micro Devices, Inc. All rights reserved.
//**************************************************************************

#include "Jacobi.hpp"

// AU_i,j = (-U_i+1,j + 2U_i,j - U_i-1,j)/dx^2 +
//          (-U_i,j+1 + 2U_i,j - U_i,j-1)/dy^2
>__global__ void LocalLaplacianKernel(const int localNx,
                                      const int localNy,
                                      const int stride,
                                      const dfloat dx,
                                      const dfloat dy,
                                      const dfloat *__restrict__ U,
                                            dfloat *__restrict__ AU) {

  const int i = threadIdx.x+blockIdx.x*blockDim.x;
  const int j = threadIdx.y+blockIdx.y*blockDim.y;

  if ((i<localNx) && (j<localNy)) {

    const int id = (i+1) + (j+1)*stride;

    const int id_l = id - 1;
    const int id_r = id + 1;
    const int id_d = id - stride;
    const int id_u = id + stride;

    AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +
             (-U[id_d] + 2*U[id] - U[id_u])/(dy*dy);
  }
}

void LocalLaplacian(grid_t& grid, mesh_t& mesh,
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./Jacobi_hip...
(gdb) b LocalLaplacianKernel
Function "LocalLaplacianKernel" not defined.
Make breakpoint pending on future shared library load? (y or [n]) y

Breakpoint 1 (LocalLaplacianKernel) pending.(gdb) run
Starting program: /home/jychang48/Downloads/hiptutorial/hip/Jacobi_hip -g 1 1
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Detaching after fork from child process 131661]
[New Thread 0x7f6c5b6c3700 (LWP 131668)]
[New Thread 0x7f6c5ad05700 (LWP 131669)]
Topology size: 1 x 1
Local domain size (current node): 4096 x 4096
Global domain size (all nodes): 4096 x 4096
[New Thread 0x7f6c58fd2700 (LWP 131670)]
Rank 0 selecting device 0 on host jychang48-workstation
[New Thread 0x7f6c5253e700 (LWP 131671)]
[Thread 0x7f6c5253e700 (LWP 131671) exited]
[New Thread 0x7f6c590ff700 (LWP 131672)]
[New Thread 0x7f6c5877f700 (LWP 131673)]
[New Thread 0x7f6c585ff700 (LWP 131674)]
Starting Jacobi run.
Iteration:   0 - Residual: 0.022108
[Switching to AMDGPU Thread 1:5:1:1 (0,0,0)/0]

Thread 9 "Jacobi_hip" hit Breakpoint 1, 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
(gdb) set scheduler-locking on
(gdb) disable 1
(gdb)
```


Rerunning rocgdb with scheduler-locking

Now we see the rest of the GPU threads.
Can also type "`thread <tid>`" to examine one particular thread

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
//**************************************************************************
//* Copyright (c) 2019, Advanced Micro Devices, Inc. All rights reserved.
//**************************************************************************

#include "Jacobi.hpp"

// AU_i,j = (-U_i+1,j + 2U_i,j - U_i-1,j)/dx^2 +
//          (-U_i,j+1 + 2U_i,j - U_i,j-1)/dy^2
__global__ void LocalLaplacianKernel(const int localNx,
                                     const int localNy,
                                     const int stride,
                                     const dfloat dx,
                                     const dfloat dy,
                                     const dfloat *__restrict__ U,
                                           dfloat *__restrict__ AU) {

-> const int i = threadIdx.x+blockIdx.x*blockDim.x;
   const int j = threadIdx.y+blockIdx.y*blockDim.y;

   if ((i<localNx) && (j<localNy)) {

     const int id = (i+1) + (j+1)*stride;

     const int id_l = id - 1;
     const int id_r = id + 1;
     const int id_d = id - stride;
     const int id_u = id + stride;

     AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +
              (-U[id_d] + 2*U[id] - U[id_u])/(dy*dy);
   }
 }

 void LocalLaplacian(grid_t& grid, mesh_t& mesh,
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
(gdb) set scheduler-locking on                                        [60/2103]
(gdb) disable 1
(gdb) step
(gdb) info threads
  Id   Target Id                                          Frame
  1    Thread 0x7f6c737088c0 (LWP 131652) "Jacobi_hip"   0x00007f6c73a95da6 in _load
er_debug_state() () from /opt/rocm/lib/libhsa-runtime64.so.1
  2    Thread 0x7f6c5b6c3700 (LWP 131668) "Jacobi_hip"   0x00007f6c74145aff in poll
() from /lib/x86_64-linux-gnu/libc.so.6
  3    Thread 0x7f6c5ad05700 (LWP 131669) "Jacobi_hip"   0x00007f6c741525ce in epoll
_wait () from /lib/x86_64-linux-gnu/libc.so.6
  4    Thread 0x7f6c58fd2700 (LWP 131670) "Jacobi_hip"   0x00007f6c7414750b in ioctl
 () from /lib/x86_64-linux-gnu/libc.so.6
  6    Thread 0x7f6c590ff700 (LWP 131672) "Jacobi_hip"   0x00007f6c758af678 in do_fu
tex_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
  7    Thread 0x7f6c5877f700 (LWP 131673) "Jacobi_hip"   0x00007f6c7414750b in ioctl
 () from /lib/x86_64-linux-gnu/libc.so.6
  8    Thread 0x7f6c585ff700 (LWP 131674) "Jacobi_hip"   0x00007f6c758af678 in do_fu
tex_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
* 9    AMDGPU Thread 1:5:1:1 (0,0,0)/0 "Jacobi_hip"      LocalLaplacianKernel (local
Nx=<optimized out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized o
ut>, dy=<optimized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:17
  10   AMDGPU Thread 1:5:1:2 (0,0,0)/1 "Jacobi_hip"      LocalLaplacianKernel (local
Nx=<optimized out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized o
ut>, dy=<optimized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
  11   AMDGPU Thread 1:5:1:3 (0,0,0)/2 "Jacobi_hip"      LocalLaplacianKernel (local
Nx=<optimized out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized o
ut>, dy=<optimized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
  12   AMDGPU Thread 1:5:1:4 (0,0,0)/3 "Jacobi_hip"      LocalLaplacianKernel (local
Nx=<optimized out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized o
ut>, dy=<optimized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
  13   AMDGPU Thread 1:5:1:5 (1,0,0)/0 "Jacobi_hip"      LocalLaplacianKernel (local
Nx=<optimized out>, localNy=<optimized out>, stride=<optimized out>, dx=<optimized o
ut>, dy=<optimized out>, U=<optimized out>, AU=<optimized out>) at Laplacian.cpp:9
  14   AMDGPU Thread 1:5:1:6 (1,0,0)/1 "Jacobi_hip"      LocalLaplacianKernel (local
```


Other tricks: export AMD_LOG_LEVEL=3

By setting the above environment variable, we can get a print of all API calls and more happening:

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
23
24      const int id_l = id - 1;
25      const int id_r = id + 1;
26      const int id_d = id - stride;
27      const int id_u = id + stride;
28
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);
31    }
32  }
33
34  void LocalLaplacian(grid_t& grid, mesh_t& 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
40 -> int localNx = mesh.Nx-2;
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);
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
Reading symbols from ./Jacobi_hip...                                       [41/144]
(gdb) b Laplacian.cpp: 40

Breakpoint 1 at 0x415080: file Laplacian.cpp, line 40.(gdb) run
Starting program: /home/jychang48/Downloads/hiptutorial/hip/Jacobi_hip -g 1 1
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Detaching after fork from child process 151584]:3:rocdevice.cpp            :457 : 4
3543864078 us: Initializing HSA stack.
:3:comgrctx.cpp              :33  : 43543879647 us: Loading COMGR library.
:3:rocdevice.cpp             :200 : 43543882744 us: Numa selects cpu agent[0]=0x55e1a
d860740(fine=0x55e1ad8490e0,coarse=0x55e1ad7e32c0, kern_arg=0x55e1ad8d7b20) for gpu
agent=0x7ffa27a13769

[New Thread 0x7f39eab37700 (LWP 151591)]
[New Thread 0x7f39ea179700 (LWP 151592)]
Topology size: 1 x 1
Local domain size (current node): 4096 x 4096
Global domain size (all nodes): 4096 x 4096
:3:rocdevice.cpp             :457 : 43544254545 us: Initializing HSA stack.
[New Thread 0x7f39e253e700 (LWP 151593)]
:3:comgrctx.cpp              :33  : 43544267424 us: Loading COMGR library.
:3:rocdevice.cpp             :200 : 43544292744 us: Numa selects cpu agent[0]=0x88ab4
0(fine=0x84b9c0,coarse=0x8a39a0, kern_arg=0x8a2900) for gpu agent=0x7f3a03a431d9
:3:hip_context.cpp           :124 : 43544298171 us: 151575: [7f3a02b7c8c0] hipInit: R
eturned hipSuccess :
:3:hip_device_runtime.cpp    :497 : 43544298229 us: 151575: [7f3a02b7c8c0] hipGetDevi
ceCount ( 0x7ffd1902c1a4 )
:3:hip_device_runtime.cpp    :499 : 43544298236 us: 151575: [7f3a02b7c8c0] hipGetDevi
ceCount: Returned hipSuccess :
:3:hip_device_runtime.cpp    :512 : 43544298264 us: 151575: [7f3a02b7c8c0] hipSetDevi
ce ( 0 )
:3:hip_device_runtime.cpp    :517 : 43544298269 us: 151575: [7f3a02b7c8c0] hipSetDevi
ce: Returned hipSuccess :
Rank 0 selecting device 0 on host jychang48-workstation
```


Other tricks: export AMD_LOG_LEVEL=3

By setting the above environment variable, we can get a print of all API calls and more happening:

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
    const int id_l = id - 1;
    const int id_r = id + 1;
    const int id_d = id - stride;
    const int id_u = id + stride;

    AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +
             (-U[id_d] + 2*U[id] - U[id_u])/(dy*dy);
  }
}

void LocalLaplacian(grid_t& grid, mesh_t& mesh,
                    hipStream_t stream,
                    dfloat* d_U,
                    dfloat* d_AU) {

  //there are (Nx-2)x(Ny-2) node on the interior of the mesh
->int localNx = mesh.Nx-2;
  int localNy = mesh.Ny-2;

  int xthreads = 16;
  int ythreads = 16;

  dim3 threads(xthreads,ythreads,1);
  dim3 blocks((localNx+xthreads-1)/xthreads,
            (localNy+ythreads-1)/ythreads, 1);

  hipLaunchKernelGGL(LocalLaplacianKernel,
                     blocks,
                     threads,
                     0, stream,
                     localNx, localNy, mesh.Nx,
                     mesh.dx, mesh.dy,
                     d_U, d_AU);
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
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 )
: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
loc ( 0x7ffd1902c700, 131072, 0 )
:3:hip_memory.cpp            :318 : 43544298589 us: 151575: [7f3a02b7c8c0] hipHostMal
loc: Returned hipSuccess : 0x7f39e8580000: duration: 116 us
:3:hip_memory.cpp            :283 : 43544298607 us: 151575: [7f3a02b7c8c0] hipMalloc
( 0x7ffd1902c708, 131072 )
:3:rocdevice.cpp             :2065: 43544298694 us: device=0x8d2890, freeMem_ = 0xfef
e0000
:3:hip_memory.cpp            :285 : 43544298702 us: 151575: [7f3a02b7c8c0] hipMalloc:
 Returned hipSuccess : 0x7f39e1a00000: duration: 95 us
:3:hip_memory.cpp            :1877: 43544298712 us: 151575: [7f3a02b7c8c0] hipMemset
( 0x7f39e1a00000, 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
 priority is: 4
[New Thread 0x7f39e19ff700 (LWP 151594)]
[Thread 0x7f39e19ff700 (LWP 151594) exited]
[New Thread 0x7f39e857f700 (LWP 151595)]
:3:rocdevice.cpp             :2638: 43544316233 us: created hardware queue 0x7f39e88c
6000 with size 1024 with priority 1, cooperative: 0
:3:devprogram.cpp            :2463: 43544489762 us: Using Code Object V4.
:3:rocvirtual.cpp            :572 : 43544495580 us: !    arg1: uint* bufUInt = ptr:0x
7f39e1a00000 obj:[0x7f39e1a00000-0x7f39e1a20000] threadId : 7f39e857f700

: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
```


Other tricks: export AMD_LOG_LEVEL=3

By setting the above environment variable, we can get a print of all API calls and more happening:

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
23
24      const int id_l = id - 1;
25      const int id_r = id + 1;
26      const int id_d = id - stride;
27      const int id_u = id + stride;
28
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);
31    }
32  }
33
34  void LocalLaplacian(grid_t& grid, mesh_t& 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
40 --> int localNx = mesh.Nx-2;
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);
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
LaunchKernel ( 0x0x949de0, 128, 1, 1, 128, 1, 1, 0, stream:0x789710, 0x7ffd[144/144]
 char array:<null>, event:0, event:0, 0, 0 )
:3:rocvirtual.cpp          :2521: 43544718510 us: [7f39e8446700]!      ShaderName :
 _Z11NormKernel1iddPKdPd

:3:hip_platform.cpp        :649 : 43544718514 us: 151575: [7f3a02b7c8c0] ihipLaunch
Kernel: Returned hipSuccess :
:3:hip_module.cpp          :433 : 43544718523 us: 151575: [7f3a02b7c8c0] hipLaunchK
ernel: Returned hipSuccess :
:3:rocvirtual.cpp          :572 : 43544718524 us: !    arg0:    = ptr:0x7f39e1a29000
 obj:[0x7f39e1a29000-0x7f39e1a29400] threadId : 7f39e8446700

:3:hip_memory.cpp          :328 : 43544718529 us: 151575: [7f3a02b7c8c0] hipMemcpy
( 0x7f39e8706000, 0x7f39e1a2a000, 8, hipMemcpyDeviceToHost )
:3:rocvirtual.cpp          :572 : 43544718534 us: !    arg1:    = ptr:0x7f39e1a2a000
 obj:[0x7f39e1a2a000-0x7f39e1a2a008] threadId : 7f39e8446700

:3:rocvirtual.cpp          :2521: 43544718546 us: [7f39e8446700]!      ShaderName :
 _Z11NormKernel2iPKdPd

:3:hip_memory.cpp          :331 : 43544719486 us: 151575: [7f3a02b7c8c0] hipMemcpy:
 Returned hipSuccess : : duration: 957 us
Iteration:    0 - Residual: 0.022108
:3:hip_device_runtime.cpp  :458 : 43544719507 us: 151575: [7f3a02b7c8c0] hipDeviceS
ynchronize (  )
:3:hip_device_runtime.cpp  :470 : 43544719512 us: 151575: [7f3a02b7c8c0] hipDeviceS
ynchronize: Returned hipSuccess :
:3:hip_event.cpp           :310 : 43544719523 us: 151575: [7f3a02b7c8c0] hipEventRe
cord ( event:0x948d00, stream:0x789710 )
:3:hip_event.cpp           :352 : 43544719530 us: 151575: [7f3a02b7c8c0] hipEventRe
cord: Returned hipSuccess :

Thread 1 "Jacobi_hip" hit Breakpoint 1, LocalLaplacian (grid=..., mesh=..., stream=0
x789710, d_U=0x7f3893e00000, d_AU=0x7f388bc00000) at Laplacian.cpp:40
(gdb)
```


Other tricks: switching between host threads

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

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```

    const int id_l = id - 1;
    const int id_r = id + 1;
    const int id_d = id - stride;
    const int id_u = id + stride;

    AU[id] = (-U[id_l] + 2*U[id] - U[id_r])/(dx*dx) +
             (-U[id_d] + 2*U[id] - U[id_u])/(dy*dy);
  }
}

void LocalLaplacian(grid_t& grid, mesh_t& mesh,
                    hipStream_t stream,
                    dfloat* d_U,
                    dfloat* d_AU) {

  //there are (Nx-2)x(Ny-2) node on the interior of the mesh
  int localNx = mesh.Nx-2;
  int localNy = mesh.Ny-2;

  int xthreads = 16;
  int ythreads = 16;

  dim3 threads(xthreads,ythreads,1);
  dim3 blocks((localNx+xthreads-1)/xthreads,
              (localNy+ythreads-1)/ythreads, 1);

  hipLaunchKernelGGL(LocalLaplacianKernel,
                     blocks,
                     threads,
                     0, stream,
                     localNx, localNy, mesh.Nx,
                     mesh.dx, mesh.dy,
                     d_U, d_AU);
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
:3:rocvirtual.cpp          :2521: 43544718546 us: [7f39e8446700]!      ShaderName :
 _Z11NormKernel2iPKdPd

:3:hip_memory.cpp          :331 : 43544719486 us: 151575: [7f3a02b7c8c0] hipMemcpy:
 Returned hipSuccess : : duration: 957 us
Iteration:    0 - Residual: 0.022108
:3:hip_device_runtime.cpp   :458 : 43544719507 us: 151575: [7f3a02b7c8c0] hipDeviceS
ynchronize (  )
:3:hip_device_runtime.cpp   :470 : 43544719512 us: 151575: [7f3a02b7c8c0] hipDeviceS
ynchronize: Returned hipSuccess :
:3:hip_event.cpp           :310 : 43544719523 us: 151575: [7f3a02b7c8c0] hipEventRe
cord ( event:0x948d00, stream:0x789710 )
:3:hip_event.cpp           :352 : 43544719530 us: 151575: [7f3a02b7c8c0] hipEventRe
cord: Returned hipSuccess :

Thread 1 "Jacobi_hip" hit Breakpoint 1, LocalLaplacian (grid=..., mesh=..., stream=0
x789710, d_U=0x7f3893e00000, d_AU=0x7f388bc00000) at Laplacian.cpp:40
(gdb) i th
  Id   Target Id                                          Frame
* 1    Thread 0x7f3a02b7c8c0 (LWP 151575) "Jacobi_hip" LocalLaplacian (grid=..., mes
h=..., stream=0x789710, d_U=0x7f3893e00000, d_AU=0x7f388bc00000) at Laplacian.cpp:40
  2    Thread 0x7f39eab37700 (LWP 151591) "Jacobi_hip" 0x00007f3a035b9aff in poll ()
 from /lib/x86_64-linux-gnu/libc.so.6
  3    Thread 0x7f39ea179700 (LWP 151592) "Jacobi_hip" 0x00007f3a035c65ce in epoll_w
ait () from /lib/x86_64-linux-gnu/libc.so.6
  4    Thread 0x7f39e253e700 (LWP 151593) "Jacobi_hip" 0x00007f3a035bb50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  6    Thread 0x7f39e857f700 (LWP 151595) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
  7    Thread 0x7f39e8446700 (LWP 151596) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
  8    Thread 0x7f39e827f700 (LWP 151597) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
(gdb)
```


Other tricks: switching between host threads

Switching between the threads gives different stack traces

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
    0x00007f3a035b9abe <+14>:   jne    0x7f3a035b9ad0 <poll+32>
    0x00007f3a035b9ac0 <+16>:   mov    $0x7,%eax
    0x00007f3a035b9ac5 <+21>:   syscall
    0x00007f3a035b9ac7 <+23>:   cmp    $0xfffffffffffff000,%rax
    0x00007f3a035b9acd <+29>:   ja     0x7f3a035b9b20 <poll+112>
    0x00007f3a035b9acf <+31>:   ret
    0x00007f3a035b9ad0 <+32>:   sub    $0x28,%rsp
    0x00007f3a035b9ad4 <+36>:   mov    %edx,0x1c(%rsp)
    0x00007f3a035b9ad8 <+40>:   mov    %rsi,0x10(%rsp)
    0x00007f3a035b9add <+45>:   mov    %rdi,0x8(%rsp)
    0x00007f3a035b9ae2 <+50>:   call   0x7f3a0353b760
    0x00007f3a035b9ae7 <+55>:   mov    0x1c(%rsp),%edx
    0x00007f3a035b9aeb <+59>:   mov    0x10(%rsp),%rsi
    0x00007f3a035b9af0 <+64>:   mov    %eax,%r8d
    0x00007f3a035b9af3 <+67>:   mov    0x8(%rsp),%rdi
    0x00007f3a035b9af8 <+72>:   mov    $0x7,%eax
    0x00007f3a035b9afd <+77>:   syscall
--> 0x00007f3a035b9aff <+79>:   cmp    $0xfffffffffffff000,%rax
    0x00007f3a035b9b05 <+85>:   ja     0x7f3a035b9b32 <poll+130>
    0x00007f3a035b9b07 <+87>:   mov    %r8d,%edi
    0x00007f3a035b9b0a <+90>:   mov    %eax,0x8(%rsp)
    0x00007f3a035b9b0e <+94>:   call   0x7f3a0353b7c0
    0x00007f3a035b9b13 <+99>:   mov    0x8(%rsp),%eax
    0x00007f3a035b9b17 <+103>:  add    $0x28,%rsp
    0x00007f3a035b9b1b <+107>:  ret
    0x00007f3a035b9b1c <+108>:  nopl   0x0(%rax)
    0x00007f3a035b9b20 <+112>:  mov    0xd5349(%rip),%rdx        # 0x7f3a0368ee70
    0x00007f3a035b9b27 <+119>:  neg    %eax
    0x00007f3a035b9b29 <+121>:  mov    %eax,%fs:(%rdx)
    0x00007f3a035b9b2c <+124>:  mov    $0xffffffff,%eax
    0x00007f3a035b9b31 <+129>:  ret
    0x00007f3a035b9b32 <+130>:  mov    0xd5337(%rip),%rdx        # 0x7f3a0368ee7
    0x00007f3a035b9b39 <+137>:  neg    %eax
    0x00007f3a035b9b3b <+139>:  mov    %eax,%fs:(%rdx)
** Dump of assembler code for function poll: (7f3a035b9ab0 - 7f3a035b9b43) **
```

```
:3:hip_memory.cpp          :331 : 43544719486 us: 151575: [7f3a02b7c8c0] hipMemcpy:
 Returned hipSuccess : : duration: 957 us
Iteration:   0 - Residual: 0.022108
:3:hip_device_runtime.cpp  :458 : 43544719507 us: 151575: [7f3a02b7c8c0] hipDeviceS
ynchronize (  )
:3:hip_device_runtime.cpp  :470 : 43544719512 us: 151575: [7f3a02b7c8c0] hipDeviceS
ynchronize: Returned hipSuccess :
:3:hip_event.cpp           :310 : 43544719523 us: 151575: [7f3a02b7c8c0] hipEventRe
cord ( event:0x948d00, stream:0x789710 )
:3:hip_event.cpp           :352 : 43544719530 us: 151575: [7f3a02b7c8c0] hipEventRe
cord: Returned hipSuccess :

Thread 1 "Jacobi_hip" hit Breakpoint 1, LocalLaplacian (grid=..., mesh=..., stream=0
x789710, d_U=0x7f3893e00000, d_AU=0x7f388bc00000) at Laplacian.cpp:40
(gdb) i th
  Id   Target Id                                   Frame
* 1    Thread 0x7f3a02b7c8c0 (LWP 151575) "Jacobi_hip" LocalLaplacian (grid=..., mes
h=..., stream=0x789710, d_U=0x7f3893e00000, d_AU=0x7f388bc00000) at Laplacian.cpp:40
  2    Thread 0x7f39eab37700 (LWP 151591) "Jacobi_hip" 0x00007f3a035b9aff in poll ()
 from /lib/x86_64-linux-gnu/libc.so.6
  3    Thread 0x7f39ea179700 (LWP 151592) "Jacobi_hip" 0x00007f3a035c65ce in epoll_w
ait () from /lib/x86_64-linux-gnu/libc.so.6
  4    Thread 0x7f39e253e700 (LWP 151593) "Jacobi_hip" 0x00007f3a035bb50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  6    Thread 0x7f39e857f700 (LWP 151595) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
  7    Thread 0x7f39e8446700 (LWP 151596) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
  8    Thread 0x7f39e827f700 (LWP 151597) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
(gdb) t 2
[Switching to thread 2 (Thread 0x7f39eab37700 (LWP 151591))]
#0  0x00007f3a035b9aff in poll () from /lib/x86_64-linux-gnu/libc.so.6
(gdb)
```


Other tricks: switching between host threads

Switching between the threads gives different stack traces

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
      0x00007f3a035c658a <+26>:   cmp    $0xfffffffffffff000,%rax
      0x00007f3a035c6590 <+32>:   ja     0x7f3a035c65f0 <epoll_wait+128>
      0x00007f3a035c6592 <+34>:   ret
      0x00007f3a035c6593 <+35>:   nopl   0x0(%rax,%rax,1)
      0x00007f3a035c6598 <+40>:   sub    $0x28,%rsp
      0x00007f3a035c659c <+44>:   mov    %edx,0x18(%rsp)
      0x00007f3a035c65a0 <+48>:   mov    %rsi,0x10(%rsp)
      0x00007f3a035c65a5 <+53>:   mov    %edi,0xc(%rsp)
      0x00007f3a035c65a9 <+57>:   mov    %ecx,0x1c(%rsp)
      0x00007f3a035c65ad <+61>:   call   0x7f3a0353b760
      0x00007f3a035c65b2 <+66>:   mov    0x1c(%rsp),%r10d
      0x00007f3a035c65b7 <+71>:   mov    0x18(%rsp),%edx
      0x00007f3a035c65bb <+75>:   mov    %eax,%r8d
      0x00007f3a035c65be <+78>:   mov    0x10(%rsp),%rsi
      0x00007f3a035c65c3 <+83>:   mov    0xc(%rsp),%edi
      0x00007f3a035c65c7 <+87>:   mov    $0xe8,%eax
      0x00007f3a035c65cc <+92>:   syscall
  --> 0x00007f3a035c65ce <+94>:   cmp    $0xfffffffffffff000,%rax
      0x00007f3a035c65d4 <+100>:  ja     0x7f3a035c6602 <epoll_wait+146>
      0x00007f3a035c65d6 <+102>:  mov    %r8d,%edi
      0x00007f3a035c65d9 <+105>:  mov    %eax,0xc(%rsp)
      0x00007f3a035c65dd <+109>:  call   0x7f3a0353b7c0
      0x00007f3a035c65e2 <+114>:  mov    0xc(%rsp),%eax
      0x00007f3a035c65e6 <+118>:  add    $0x28,%rsp
      0x00007f3a035c65ea <+122>:  ret
      0x00007f3a035c65eb <+123>:  nopl   0x0(%rax,%rax,1)
      0x00007f3a035c65f0 <+128>:  mov    0xc8879(%rip),%rdx        # 0x7f3a0368ee70
      0x00007f3a035c65f7 <+135>:  neg    %eax
      0x00007f3a035c65f9 <+137>:  mov    %eax,%fs:(%rdx)
      0x00007f3a035c65fc <+140>:  mov    $0xffffffff,%eax
      0x00007f3a035c6601 <+145>:  ret
      0x00007f3a035c6602 <+146>:  mov    0xc8867(%rip),%rdx        # 0x7f3a0368ee7
      0x00007f3a035c6609 <+153>:  neg    %eax
      0x00007f3a035c660b <+155>:  mov    %eax,%fs:(%rdx)
** Dump of assembler code for function epoll_wait: (7f3a035c6570 - 7f3a035c6613) **
```

```
Iteration:    0 - Residual: 0.022108
:3:hip_device_runtime.cpp   :458 : 43544719507 us: 151575: [7f3a02b7c8c0] hipDeviceS
ynchronize (  )
:3:hip_device_runtime.cpp   :470 : 43544719512 us: 151575: [7f3a02b7c8c0] hipDeviceS
ynchronize: Returned hipSuccess :
:3:hip_event.cpp            :310 : 43544719523 us: 151575: [7f3a02b7c8c0] hipEventRe
cord ( event:0x948d00, stream:0x789710 )
:3:hip_event.cpp            :352 : 43544719530 us: 151575: [7f3a02b7c8c0] hipEventRe
cord: Returned hipSuccess :

Thread 1 "Jacobi_hip" hit Breakpoint 1, LocalLaplacian (grid=..., mesh=..., stream=0
x789710, d_U=0x7f3893e00000, d_AU=0x7f388bc00000) at Laplacian.cpp:40
(gdb) i th
  Id   Target Id                                  Frame
* 1    Thread 0x7f3a02b7c8c0 (LWP 151575) "Jacobi_hip" LocalLaplacian (grid=..., mes
h=..., stream=0x789710, d_U=0x7f3893e00000, d_AU=0x7f388bc00000) at Laplacian.cpp:40
  2    Thread 0x7f39eab37700 (LWP 151591) "Jacobi_hip" 0x00007f3a035b9aff in poll ()
 from /lib/x86_64-linux-gnu/libc.so.6
  3    Thread 0x7f39ea179700 (LWP 151592) "Jacobi_hip" 0x00007f3a035c65ce in epoll_w
ait () from /lib/x86_64-linux-gnu/libc.so.6
  4    Thread 0x7f39e253e700 (LWP 151593) "Jacobi_hip" 0x00007f3a035bb50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  6    Thread 0x7f39e857f700 (LWP 151595) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
  7    Thread 0x7f39e8446700 (LWP 151596) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
  8    Thread 0x7f39e827f700 (LWP 151597) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
(gdb) t 2
[Switching to thread 2 (Thread 0x7f39eab37700 (LWP 151591))]
#0  0x00007f3a035b9aff in poll () from /lib/x86_64-linux-gnu/libc.so.6
(gdb) t 3
[Switching to thread 3 (Thread 0x7f39ea179700 (LWP 151592))]
#0  0x00007f3a035c65ce in epoll_wait () from /lib/x86_64-linux-gnu/libc.so.6
(gdb)
```


Other tricks: switching between host threads

Switching between the threads gives different stack traces

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
Dump of assembler code for function ioctl:
    0x00007f3a035bb500 <+0>:     endbr64
    0x00007f3a035bb504 <+4>:     mov    $0x10,%eax
    0x00007f3a035bb509 <+9>:     syscall
--> 0x00007f3a035bb50b <+11>:    cmp    $0xfffffffffffff001,%rax
    0x00007f3a035bb511 <+17>:    jae    0x7f3a035bb514 <ioctl+20>
    0x00007f3a035bb513 <+19>:    ret
    0x00007f3a035bb514 <+20>:    mov    0xd3955(%rip),%rcx        # 0x7f3a0368ee70
    0x00007f3a035bb51b <+27>:    neg    %eax
    0x00007f3a035bb51d <+29>:    mov    %eax,%fs:(%rcx)
    0x00007f3a035bb520 <+32>:    or     $0xffffffffffffffff,%rax
    0x00007f3a035bb524 <+36>:    ret
End of assembler dump.
** Dump of assembler code for function ioctl: (7f3a035bb500 - 7f3a035bb524) **
```

```
:3:hip_device_runtime.cpp   :470 : 43544719512 us: 151575: [7f3a02b7c8c0] hipDeviceS
ynchronize: Returned hipSuccess :
:3:hip_event.cpp            :310 : 43544719523 us: 151575: [7f3a02b7c8c0] hipEventRe
cord ( event:0x948d00, stream:0x789710 )
:3:hip_event.cpp            :352 : 43544719530 us: 151575: [7f3a02b7c8c0] hipEventRe
cord: Returned hipSuccess :

Thread 1 "Jacobi_hip" hit Breakpoint 1, LocalLaplacian (grid=..., mesh=..., stream=0
x789710, d_U=0x7f3893e00000, d_AU=0x7f388bc00000) at Laplacian.cpp:40
(gdb) i th
  Id   Target Id                                   Frame
* 1    Thread 0x7f3a02b7c8c0 (LWP 151575) "Jacobi_hip" LocalLaplacian (grid=..., mes
h=..., stream=0x789710, d_U=0x7f3893e00000, d_AU=0x7f388bc00000) at Laplacian.cpp:40
  2    Thread 0x7f39eab37700 (LWP 151591) "Jacobi_hip" 0x00007f3a035b9aff in poll ()
 from /lib/x86_64-linux-gnu/libc.so.6
  3    Thread 0x7f39ea179700 (LWP 151592) "Jacobi_hip" 0x00007f3a035c65ce in epoll_w
ait () from /lib/x86_64-linux-gnu/libc.so.6
  4    Thread 0x7f39e253e700 (LWP 151593) "Jacobi_hip" 0x00007f3a035bb50b in ioctl (
) from /lib/x86_64-linux-gnu/libc.so.6
  6    Thread 0x7f39e857f700 (LWP 151595) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
  7    Thread 0x7f39e8446700 (LWP 151596) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
  8    Thread 0x7f39e827f700 (LWP 151597) "Jacobi_hip" 0x00007f3a04d23678 in do_fute
x_wait.constprop () from /lib/x86_64-linux-gnu/libpthread.so.0
(gdb) t 2
[Switching to thread 2 (Thread 0x7f39eab37700 (LWP 151591))]
#0  0x00007f3a035b9aff in poll () from /lib/x86_64-linux-gnu/libc.so.6
(gdb) t 3
[Switching to thread 3 (Thread 0x7f39ea179700 (LWP 151592))]
#0  0x00007f3a035c65ce in epoll_wait () from /lib/x86_64-linux-gnu/libc.so.6
(gdb) t 4
[Switching to thread 4 (Thread 0x7f39e253e700 (LWP 151593))]
#0  0x00007f3a035bb50b in ioctl () from /lib/x86_64-linux-gnu/libc.so.6
(gdb)
```


AMD

What to do if your program returns an error?

In a perfect world, all code should run successfully like this …

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
jychang48@jychang48-workstation:~/Downloads/hiptutorial/hip$ ./Jacobi_hip -g 1 1
Topology size: 1 x 1
Local domain size (current node): 4096 x 4096
Global domain size (all nodes): 4096 x 4096
Rank 0 selecting device 0 on host jychang48-workstation
Starting Jacobi run.
Iteration:   0 - Residual: 0.022108
Iteration: 100 - Residual: 0.000625
Iteration: 200 - Residual: 0.000371
Iteration: 300 - Residual: 0.000274
Iteration: 400 - Residual: 0.000221
Iteration: 500 - Residual: 0.000187
Iteration: 600 - Residual: 0.000163
Iteration: 700 - Residual: 0.000145
Iteration: 800 - Residual: 0.000131
Iteration: 900 - Residual: 0.000120
Iteration: 1000 - Residual: 0.000111
Stopped after 1000 iterations with residue 0.000111
Total Jacobi run time: 2.0454 sec.
Measured lattice updates: 8.20 GLU/s (total), 8.20 GLU/s (per process)
Measured FLOPS: 139.44 GFLOPS (total), 139.44 GFLOPS (per process)
Measured device bandwidth: 787.42 GB/s (total), 787.42 GB/s (per process)
jychang48@jychang48-workstation:~/Downloads/hiptutorial/hip$
```


What to do if your program returns an error?

… but every so often we come across GPU errors like this

```
jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip
jychang48@jychang48-workstation:~/Downloads/hiptutorial/hip$ ./Jacobi_hip -g 1 1
Topology size: 1 x 1
Local domain size (current node): 4096 x 4096
Global domain size (all nodes): 4096 x 4096
Rank 0 selecting device 0 on host jychang48-workstation
Starting Jacobi run.
Iteration:   0 - Residual: 0.022108
Memory access fault by GPU node-1 (Agent handle: 0x121b750) on address 0x7f2c6fe00000. Reason: Page not present or supervisor privilege.
[jychang48-workstation:133929] *** Process received signal ***
[jychang48-workstation:133929] Signal: Aborted (6)
[jychang48-workstation:133929] Signal code:  (-6)
[jychang48-workstation:133929] [ 0] /lib/x86_64-linux-gnu/libpthread.so.0(+0x15420)[0x7f2dd84ea420]
[jychang48-workstation:133929] [ 1] /lib/x86_64-linux-gnu/libc.so.6(gsignal+0xcb)[0x7f2dd6caf18b]
[jychang48-workstation:133929] [ 2] /lib/x86_64-linux-gnu/libc.so.6(abort+0x12b)[0x7f2dd6c8e859]
[jychang48-workstation:133929] [ 3] /opt/rocm/lib/libhsa-runtime64.so.1(+0x6933f)[0x7f2dd66c233f]
[jychang48-workstation:133929] [ 4] /opt/rocm/lib/libhsa-runtime64.so.1(+0x6b4cb)[0x7f2dd66c44cb]
[jychang48-workstation:133929] [ 5] /opt/rocm/lib/libhsa-runtime64.so.1(+0x16467)[0x7f2dd666f467]
[jychang48-workstation:133929] [ 6] /lib/x86_64-linux-gnu/libpthread.so.0(+0x9609)[0x7f2dd84de609]
[jychang48-workstation:133929] [ 7] /lib/x86_64-linux-gnu/libc.so.6(clone+0x43)[0x7f2dd6d8b293]
[jychang48-workstation:133929] *** End of error message ***
Aborted (core dumped)
jychang48@jychang48-workstation:~/Downloads/hiptutorial/hip$
```


What to do if your program returns an error?

Launch rocgdb and invoke the options shown in the picture:

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```cpp
//**************************************************************************
//* Copyright (c) 2019, Advanced Micro Devices, Inc. All rights reserved.
//**************************************************************************

#include "Jacobi.hpp"

/**
 * @file JacobiMain.cpp
 * @brief This contains the application entry point
 */

/**
 * @brief The application entry point
 *
 * @param[in] argc  The number of command-line arguments
 * @param[in] argv  The list of command-line arguments
 */
int main(int argc, char ** argv)
{
  MPI_Init(&argc, &argv);

  MPI_Comm comm = MPI_COMM_WORLD;

  grid_t grid;
  mesh_t mesh;

  // Extract topology and domain dimensions from the command-line arguments
  ParseCommandLineArguments(argc, argv,
                            comm,
                            grid,
                            mesh);

  Jacobi_t Jacobi(grid, mesh);

/home/jychang48/Downloads/hiptutorial/hip/JacobiMain.cpp
```

```
GNU gdb (rocm-rel-4.2-21) 10.1
Copyright (C) 2020 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 ./Jacobi_hip...
(gdb) set pagination off
(gdb) set non-stop on
(gdb) b abort
Function "abort" not defined.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (abort) pending.
(gdb)
```

What to do if your program returns an error?

We now see the offending line which invokes the error. Uncommenting out lines 20 and 31 will fix this particular issue.

jychang48@jychang48-workstation: ~/Downloads/hiptutorial/hip

```
12                                  const dfloat dx,
13                                  const dfloat dy,
14                                  const dfloat *__restrict__ U,
15                                        dfloat *__restrict__ AU) {
16
17   const int i = threadIdx.x+blockIdx.x*blockDim.x;
18   const int j = threadIdx.y+blockIdx.y*blockDim.y;
19
20 //  if ((i<localNx) && (j<localNy)) {
21
22     const int id = (i+1) + (j+1)*stride;
23
24     const int id_l = id - 1;
25     const int id_r = id + 1;
26     const int id_d = id - stride;
27     const int id_u = id + stride;
28
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);
31 //  }
32 }
33
34 void LocalLaplacian(grid_t& grid, mesh_t& 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
40   int localNx = mesh.Nx-2;
41   int localNy = mesh.Ny-2;
42
43   int xthreads = 16;
44   int ythreads = 16;
45
/home/jychang48/Downloads/hiptutorial/hip/Laplacian.cpp
```

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./Jacobi_hip...
(gdb) set pagination off
(gdb) set non-stop on
(gdb) b abort
Function "abort" not defined.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (abort) pending.
(gdb) run
Starting program: /home/jychang48/Downloads/hiptutorial/hip/Jacobi_hip -g 1 1
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Detaching after fork from child process 134151]
[New Thread 0x7f3d1d486700 (LWP 134158)]
[New Thread 0x7f3d1cac8700 (LWP 134159)]
Topology size: 1 x 1
Local domain size (current node): 4096 x 4096
Global domain size (all nodes): 4096 x 4096
[New Thread 0x7f3d152d0700 (LWP 134160)]
Rank 0 selecting device 0 on host jychang48-workstation
[New Thread 0x7f3c0506d700 (LWP 134161)]
[Thread 0x7f3c0506d700 (LWP 134161) exited]
[New Thread 0x7f3d15544700 (LWP 134162)]
[New Thread 0x7f3d153d2700 (LWP 134170)]
[New Thread 0x7f3d148ff700 (LWP 134171)]
Starting Jacobi run.
Iteration:   0 - Residual: 0.022108
Memory access fault by GPU node-1 (Agent handle: 0x12bf5a0) on address 0x7f3bcfe0000
0. Reason: Page not present or supervisor privilege.

Thread 4 "Jacobi_hip" hit Breakpoint 1, 0x00007f3d35e1872e in abort () from /lib/x86
_64-linux-gnu/libc.so.6
(gdb)
```


And more…

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

- https://github.com/RadeonOpenCompute/ROCm/blob/master/ROCm_Debugger_User_Guide_v4.2.pdf
- `/opt/rocm-4.2.0/share/doc/rocgdb/rocannotate.pdf`
- `/opt/rocm-4.2.0/share/doc/rocgdb/rocgdb.pdf`
- `/opt/rocm-4.2.0/share/doc/rocgdb/rocrefcard.pdf`
- `/opt/rocm-4.2.0/share/doc/rocgdb/rocstabs.pdf`

Part 2: Math Libraries

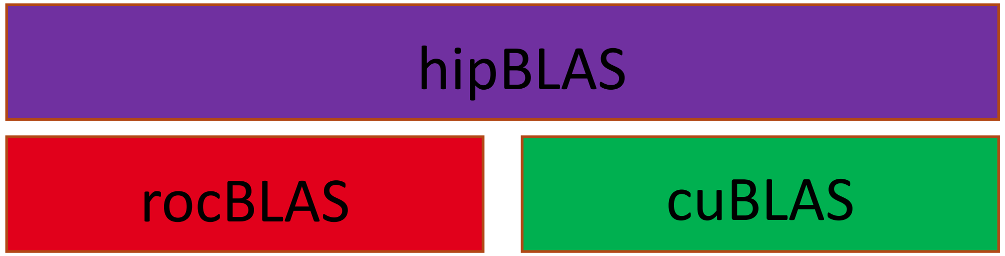
Decoder ring: Math library equivalents

CUBLAS	**ROCBLAS**	Basic Linear Algebra Subroutines
CUFFT	**ROCFFT**	Fast Fourier Transforms
THRUST	**ROCTHRUST**	Deep Learning Library
CUB	**ROCPRIM**	Optimized Parallel Primitives
EIGEN	**EIGEN**	C++ Template Library for Linear Algebra

MORE INFO AT: GITHUB.COM/ROCM-DEVELOPER-TOOLS/HIP → HIP_PORTING_GUIDE.MD

AMD GPU Math Libraries

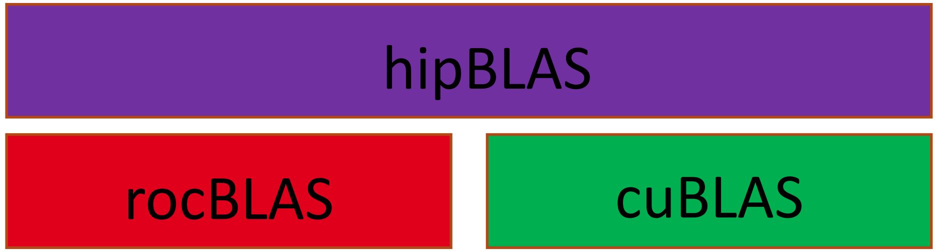
- A note on naming conventions:
  - roc* -> AMGCN library usually written in HIP
  - cu* -> NVIDIA PTX libraries
  - hip* -> usually interface layer on top of roc*/cu* backends
- hip* libraries:
  - Can be compiled by hipcc and can generate a call for the device you have:
    - hipcc->clang->AMD GCN ISA
    - hipcc->nvcc (inlined)->NVPTX
  - Just a thin wrapper that marshals calls off to a “backend” library:
    - corresponding roc* library backend containing optimized GCN
    - corresponding cu* library backend containing NVPTX for NVIDIA devices
  - E.g., hipBLAS is a marshalling library:



AMD

AMD GPU Libraries: BLAS

- rocBLAS – `sudo apt install rocblas`
  - Source code: https://github.com/ROCmSoftwarePlatform/rocBLAS
  - Documentation: https://rocblas.readthedocs.io/en/latest/
  - Basic linear algebra functionality
    - axpy, gemv, trsm, etc
  - Use hipBLAS if you need portability between AMD and NVIDIA devices
- hipBLAS - `sudo apt install hipblas`
  - Documentation: https://github.com/ROCmSoftwarePlatform/hipBLAS/wiki/Exported-functions
  - Use this if you need portability between AMD and NVIDIA
  - It is just a thin wrapper:
    - It can dispatch calls to rocBLAS for AMD devices
    - It can dispatch calls to cuBLAS for NVIDIA devices



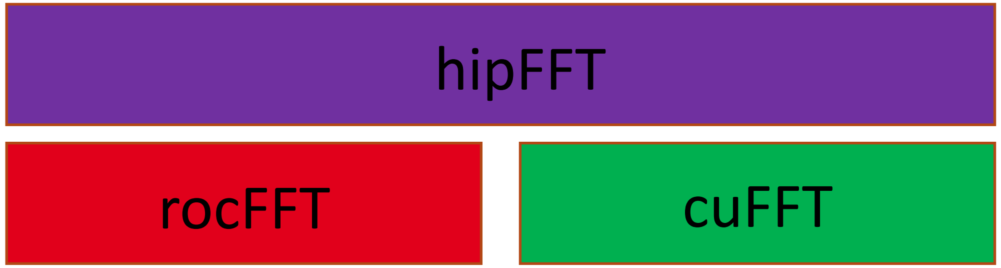
AMD GPU Libraries: rocBLAS example

- rocBLAS
  - Documentation: https://rocblas.readthedocs.io/en/latest/
  - Level 1, 2, and 3 functionality
    - axpy, gemv, trsm, etc
  - Note: rocBLAS syntax matches BLAS closer than hipBLAS or cuBLAS
    - Use hipBLAS only if you need portability between AMD and NVIDIA devices
  - Link with: `-lrocblas`

```
#include <rocblas.h>

int main(int argc, char ** argv) {
  rocblas_int N = 500000;

  // Allocate device memory
  double * dx, * dy;
  hipMalloc(&dx, sizeof(double) * N);
  hipMalloc(&dy, sizeof(double) * N);

  // Allocate host memory (and fill up the arrays) here
  std::vector<double> hx(N), hy(N);

  // Copy host arrays to device
  hipMemcpy(dx, hx.data(), sizeof(double) * N, hipMemcpyHostToDevice);
  hipMemcpy(dy, hy.data(), sizeof(double) * N, hipMemcpyHostToDevice);

  const double alpha = 1.0;
  rocblas_handle handle;
  rocblas_create_handle(&handle);
  rocblas_status status;
  status = rocblas_daxpy(handle, N, &alpha, dx, 1, dy, 1);
  rocblas_destroy_handle(handle);

  // Copy result back to host
  hipMemcpy(hy.data(), dy, sizeof(double) * N, hipMemcpyDeviceToHost);
  hipFree(dx);
  hipFree(dy);
  return 0;
}
```


AMD GPU Libraries: FFT

- rocFFT – `sudo apt install rocfft`
  - Source code: https://github.com/ROCmSoftwarePlatform/rocFFT
  - Documentation: https://rocfft.readthedocs.io/en/latest/
  - Implementation of Discrete Fourier Transforms (DFT), leveraging mathematical symmetries to reduce algorithmic complexity from $O(N^2)$ to $O(N \log N)$
  - Use hipFFT (`sudo apt install hipfft`) if you need portability between AMD and NVIDIA devices
- Basic steps in rocFFT:
  1. Initialize the library by calling `rocfft_setup()`
  2. Create a plan for the FFT needed
  3. Optionally allocate a work buffer
  4. Execute the plan
  5. Free the work buffer if needed
  6. Destroy the plan
  7. Terminate the library by calling `rocfft_cleanup()`



AMD

AMD GPU Libraries: rocFFT example

```
#include <iostream>
#include <vector>
#include "hip/hip_runtime_api.h"
#include "hip/hip_vector_types.h"
#include "rocfft.h"

int main()
{
  rocfft_setup();

  // Create HIP device buffer
  size_t N = 16;
  size_t Nbytes = N * sizeof(float2);
  float2 *x;
  hipMalloc(&x, Nbytes);

  // Initialize data
  std::vector<float2> cx(N);
  for (size_t i = 0; i < N; i++) {
    cx[i].x = 1; cx[i].y = -1;
  }

  //  Copy data to device
  hipMemcpy(x, cx.data(), Nbytes, hipMemcpyHostToDevice);

  // Create rocFFT plan
  rocfft_plan plan = nullptr;
  size_t length = N;
  rocfft_plan_create(&plan, rocfft_placement_inplace,
     rocfft_transform_type_complex_forward, rocfft_precision_single,
     1, &length, 1, nullptr);

  // Check if the plan requires a work buffer
  size_t work_buf_size = 0;
  rocfft_plan_get_work_buffer_size(plan, &work_buf_size);
  void* work_buf = nullptr;
  rocfft_execution_info info = nullptr;
  if(work_buf_size) {
    rocfft_execution_info_create(&info);
    hipMalloc(&work_buf, work_buf_size);
    rocfft_execution_info_set_work_buffer(info, work_buf, work_buf_size);
  }

  // Execute plan
  rocfft_execute(plan, (void**) &x, nullptr, info);
  hipDeviceSynchronize();

  // Clean up work buffer
  if(work_buf_size) {
    hipFree(work_buf);
    rocfft_execution_info_destroy(info);
  }

  // Destroy plan
  rocfft_plan_destroy(plan);

  // Copy result back to host
  std::vector<float2> y(N);
  hipMemcpy(y.data(), x, Nbytes, hipMemcpyDeviceToHost);

  // Free device buffer
  hipFree(x);
  rocfft_cleanup();
  return 0;
}
```


Some Links to Key Libraries

- BLAS
  - rocBLAS (https://github.com/ROCmSoftwarePlatform/rocBLAS)
  - hipBLAS (https://github.com/ROCmSoftwarePlatform/hipBLAS)
- FFTs
  - rocFFT (https://github.com/ROCmSoftwarePlatform/rocFFT)
- Random number generation
  - rocRAND (https://github.com/ROCmSoftwarePlatform/rocRAND)
  - hipRAND (https://github.com/ROCmSoftwarePlatform/hipRAND)
- Sparse linear algebra
  - rocSPARSE (https://github.com/ROCmSoftwarePlatform/rocSPARSE)
  - hipSPARSE (https://github.com/ROCmSoftwarePlatform/hipSPARSE)
- Iterative solvers
  - rocALUTION (https://github.com/ROCmSoftwarePlatform/rocALUTION)
- Parallel primitives
  - rocPRIM (https://github.com/ROCmSoftwarePlatform/rocPRIM)
  - hipCUB (https://github.com/ROCmSoftwarePlatform/hipCUB)

More links to key libraries

Machine Learning libraries and Frameworks

- Tensorflow: https://github.com/ROCmSoftwarePlatform/tensorflow-upstream
- Pytorch: https://github.com/ROCmSoftwarePlatform/pytorch
- MIOpen (similar to cuDNN): https://github.com/ROCmSoftwarePlatform/MIOpen
- Tensile: https://github.com/ROCmSoftwarePlatform/Tensile
- RCCL (ROCm analogue of NCCL): https://github.com/ROCmSoftwarePlatform/rccl

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.

AMD