Large Scale Debugging
Project Meeting Report - December 2016

Didier Nadeau
Under the supervision of Michel Dagenais

Distributed Open Reliable Systems Analysis Lab
École Polytechnique de Montréal
Table of contents

1. AMD HSA

2. GDB
   - AMD HSA Modification
   - MI Python extension

3. Eclipse CDT
   - Stack Aggregation View
   - HSA Debug View
   - GPU Visualizer
   - Future Work
The GPUOpen Initiative

GPUOpen

An initiative launched in 2015 by AMD to provide an open-source software stack to interact with graphic cards for professional use and personal use.

Heterogeneous System Architecture (HSA) Foundation

- Provide a standardized interface for programmer
- Multiple instruction sets
- Radeon Open Compute is an implementation by AMD
Improvements were made on AMD’s gdb version for HSA debugging.

**Modifications to amd-gdb**

- Usage of breakpoints instead of signals
- Outputting hsa information in machine interface format
- New command to add hsa specific breakpoint
Python interface in GDB

GDB allows users to easily modify its behaviour and implement new functionalities in python.

Partial feature list

- API for threads, breakpoints, inferiors, etc.
- Handling various events raised by GDB
- Customized printing
- Implementing new commands
Implementing new commands

It can be done by extending GDB Command class:

```python
#!/usr/bin/python
import gdb

class TestCommand(gdb.Command):
    def __init__(self):
        super(TestCommand, self).__init__("test-command", gdb.COMMAND_USER)

    def invoke(self, self, arg, from_tty):
        print("A new GDB command is defined")

TestCommand()
```

However, this does not integrate well with an IDE
Machine Interface Commands

A new python module for GDB, MICommand, has been created. It facilitates interfacing python command with an IDE.

- Recognized as standard MI commands
- Uses standard Machine Interface syntax
- Access to the Python API already in place
Debug View

Standard Debug View

The debug view is the tree view to show and select the various processes, threads and stack.

```
Dragonizer [C/C++ Application]
  dragonizer [26788] [cores: 1,2,6,7]
    Thread #1 [dragonizer] 26788 [core: 6] (Suspended : Container)
    Thread #8 [dragonizer] 26821 [core: 1] (Suspended : Container)
    Thread #9 [dragonizer] 26822 [core: 7] (Suspended : Container)
    Thread #10 [dragonizer] 26823 [core: 6] (Suspended : Container)
    Thread #11 [dragonizer] 26824 [core: 7] (Suspended : Container)
    Thread #12 [dragonizer] 26825 [core: 7] (Suspended : Breakpoint)
    Thread #13 [dragonizer] 26826 [core: 2] (Suspended : Container)
  gdb (7.7.1)
```
Stack Aggregation View

Proposed view

This debug view merge the call stack of each thread to display useful information when there is a large number of threads.
Integrating HSA to the debug view

Waves

The wave concept can be related to a thread: each wave has a program counter and executes the same instructions on a data group, which is similar to a CPU thread using vector instructions.

Challenge

A single GPU can support an enormous amount of waves executing at the same time. The AMD R9 Nano used for the test can support up to a maximum of 2560 waves.
Adapted debug view

First iteration of the HSA debug view: grouping by work-group.

clone() at 0x7fff703f37d

/opt/rocm/gdb/bin/x86_64/rocm-gdb (7.8)

HSA WorkGroup (0,x,x)

Wave 0x40802dc0, Work-Group (0,1,0)
Wave 0x40802fd0, Work-Group (0,2,0)
Wave 0x40802bc0, Work-Group (0,0,0)
Wave 0x408024c0, Work-Group (0,4,0)
Wave 0x408022c0, Work-Group (0,3,0)
Wave 0x40802fe0, Work-Group (0,2,0)
Wave 0x40802ff0, Work-Group (0,2,0)
Wave 0x40802dd0, Work-Group (0,1,0)
Wave 0x40802de0, Work-Group (0,1,0)
GPU Visualizer

**Visualizer**

Adaptation of the multicore visualizer available in Eclipse CDT to display GPU structure and waves.

**Java FX**

The visualizer has been modified to use the new version of the Graphical Eclipse Framework based on Java FX.
GPU Visualizer
Future Work

- Optimize the various views
- Integrate the HSA views together
- Add a focus (Work-group, compute-unit, etc) to HSA debugging
- Submit work to GDB, CDT
Any Questions?

Contact
didier.nadeau@polymtl.ca