Large Scale Debugging

Project Meeting Report - May 2016

Didier Nadeau
Under the supervision of Michel Dagenais

Distributed Open Reliable Systems Analysis Lab
École Polytechnique de Montréal

Table of contents

Multicore debugging

Challenges of parallel debugging

- Scalability to 100-1000s cores
- Ease of use of available commands
- Efficient data collection with dynamic tracepoints
- Conditional and thread-specific breakpoints
- Minimal perturbation of debuggee

Tracing with GDB

Normal tracepoint

The standard tracing mode uses interruptions. The debugger collects information and resumes execution. The overhead is very large, possibly more than 100 μs per breakpoint.

Fast tracepoint

Implemented in the debuggee memory space using a jump instruction and displaced code.

Insertion of a dynamic jump

```
0000000000400b2d <do_ops>:
  400b2d:       55                      push   %rbp
  400b2e:       48 89 e5                mov    %rsp,%rbp
  400b31:       f2 0f 11 45 e8          movsd  %xmm0,-0x18(%rbp)
  400b36:       89 7d e4                mov    %edi,-0x1c(%rbp)
  400b39:       b8 00 00 00 00          mov    $0x0,%eax
  400b3e:       48 89 45 f8             mov    %rax,-0x8(%rbp)
  400b42:       c7 45 f4 00 00 00 00    movl   $0x0,-0xc(%rbp)
  400b49:       eb 4b                   jmp    400b96 <do_ops+0x69>
  400b4b:       f2 0f 2a 45 f4          cvtsi2sdl -0xc(%rbp),%xmm0
  400b50:       f2 0f 10 4d e8          movsd  -0x18(%rbp),%xmm1
  400b55:       f2 0f 58 c1             addsd  %xmm1,%xmm0
  400b59:       f2 0f 11 45 e8          movsd  %xmm0,-0x18(%rbp)
  400b5e:       f2 0f 10 45 e8          movsd  -0x18(%rbp),%xmm0
  400b63:       f2 0f 10 0d 95 0c 00    movsd  0xc95(%rip),%xmm1
  400b6a:       00
  400b6b:       f2 0f 5c c1             subsd  %xmm1,%xmm0
  400b6f:       f2 0f 10 55 e8          movsd  -0x18(%rbp),%xmm2
  400b74:       f2 0f 10 0d 84 0c 00    movsd  0xc84(%rip),%xmm1
  400b7b:       00
  400b7c:       f2 0f 58 ca             addsd  %xmm2,%xmm1
  400b80:       f2 0f 5e c1             divsd  %xmm1,%xmm0
  400b84:       f2 0f 10 4d f8          movsd  -0x8(%rbp),%xmm1
  400b89:       f2 0f 58 c1             addsd  %xmm1,%xmm0
  400b8d:       f2 0f 11 45 f8          movsd  %xmm0,-0x8(%rbp)
  400b92:       83 45 f4 01             addl   $0x1,-0xc(%rbp)
  400b96:       8b 45 f4                mov    -0xc(%rbp),%eax
  400b99:       3b 45 e4                cmp    -0x1c(%rbp),%eax
  400b9c:       7c ad                   jl     400b4b <do_ops+0x1e>
  400b9e:       48 8b 45 f8             mov    -0x8(%rbp),%rax
  400ba2:       48 89 45 d8             mov    %rax,-0x28(%rbp)
  400ba6:       f2 0f 10 45 d8          movsd  -0x28(%rbp),%xmm0
  400bab:       5d                      pop    %rbp
  400bac:       c3                      retq
```

```
0000000000400b2d <do_ops>:
  400b2d:       55                      push   %rbp
  400b2e:       48 89 e5                mov    %rsp,%rbp
  400b31:       f2 0f 11 45 e8          movsd  %xmm0,-0x18(%rbp)
  400b36:       89 7d e4                mov    %edi,-0x1c(%rbp)
  400b39:       b8 00 00 00 00          mov    $0x0,%eax
  400b3e:       48 89 45 f8             mov    %rax,-0x8(%rbp)
  400b42:       c7 45 f4 00 00 00 00    movl   $0x0,-0xc(%rbp)
  400b49:       eb 4b                   jmp    400b96 <do_ops+0x69>
  400b4b:       f2 0f 2a 45 f4          cvtsi2sdl -0xc(%rbp),%xmm0
  400b50:       f2 0f 10 4d e8          movsd  -0x18(%rbp),%xmm1
  400b55:       f2 0f 58 c1             addsd  %xmm1,%xmm0
  400b59:       f2 0f 11 45 e8          movsd  %xmm0,-0x18(%rbp)
  400b5e:       f2 0f 10 45 e8          movsd  -0x18(%rbp),%xmm0
  400b63:       f2 0f 10 0d 95 0c 00    movsd  0xc95(%rip),%xmm1
  400b6a:       00
  400b6b:       f2 0f 5c c1             subsd  %xmm1,%xmm0
  400b6f:       f2 0f 10 55 e8          movsd  -0x18(%rbp),%xmm2
  400b74:       f2 0f 10 0d 84 0c 00    movsd  0xc84(%rip),%xmm1
  400b7b:       00
  400b7c:       f2 0f 58 ca             addsd  %xmm2,%xmm1
  400b80:       f2 0f 5e c1             divsd  %xmm1,%xmm0
  400b84:       e9 60 ff  ff  ff        jmpq   400910
  400b89:       f2 0f 58 c1             addsd  %xmm1,%xmm0
  400b8d:       f2 0f 11 45 f8          movsd  %xmm0,-0x8(%rbp)
  400b92:       83 45 f4 01             addl   $0x1,-0xc(%rbp)
  400b96:       8b 45 f4                mov    -0xc(%rbp),%eax
  400b99:       3b 45 e4                cmp    -0x1c(%rbp),%eax
  400b9c:       7c ad                   jl     400b4b <do_ops+0x1e>
  400b9e:       48 8b 45 f8             mov    -0x8(%rbp),%rax
  400ba2:       48 89 45 d8             mov    %rax,-0x28(%rbp)
  400ba6:       f2 0f 10 45 d8          movsd  -0x28(%rbp),%xmm0
  400bab:       5d                      pop    %rbp
  400bac:       c3                      retq
```

Replacement of a 5 bytes instruction with a jump to the tracing function

Implementation limits

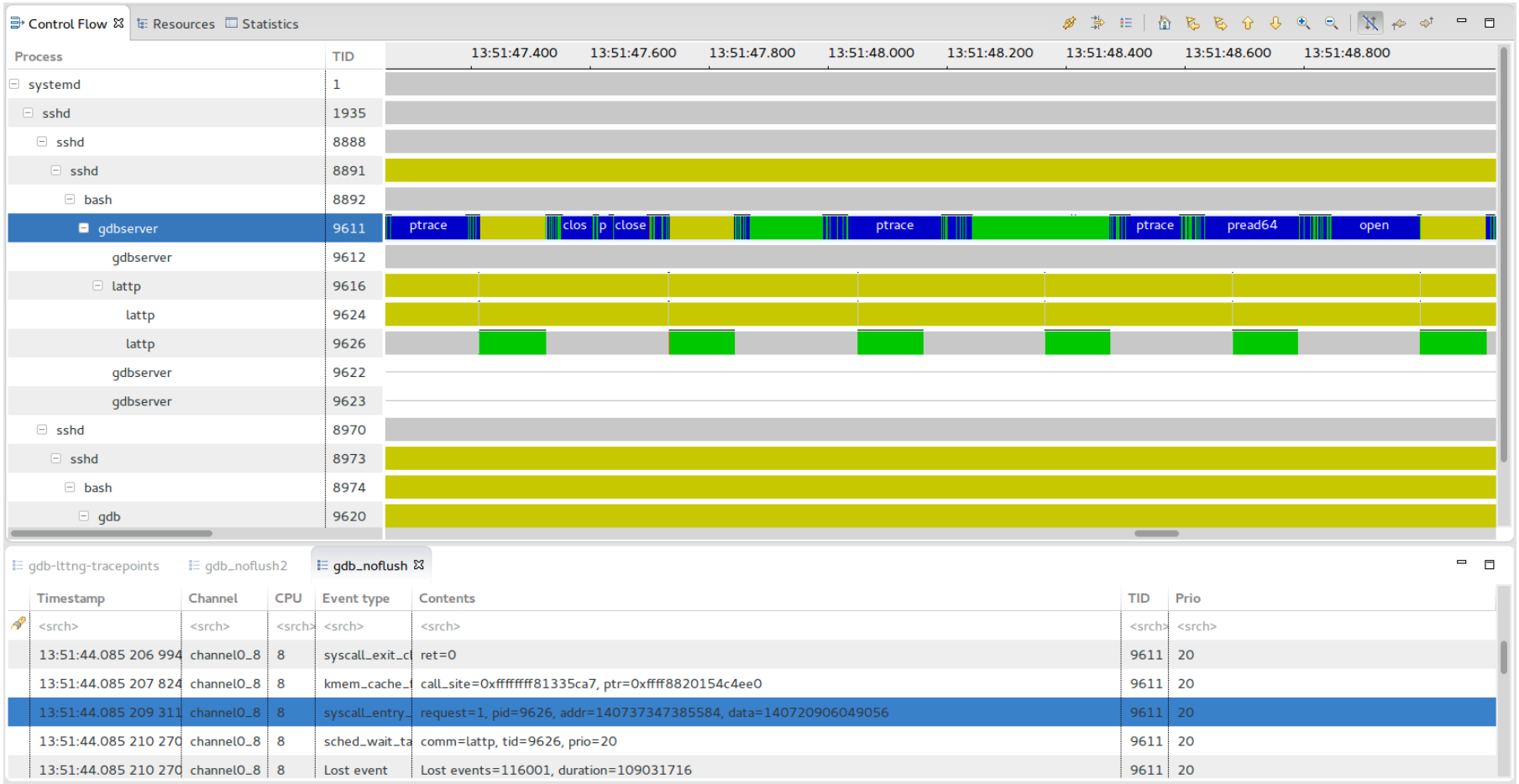
Use a single buffer shared by every threads

- No concurrent data collection
- Global spinlock in the jump pad

GDBServer handles data transfer

- Stops the program to empty the buffer
- GDBserver must handle the data as well as normal debugging operations

Flushing of the agent's buffer



GDBServer copies the buffer content into its own memory and restarts the thread

Dynamic insertion of LTTng UST tracepoints

GDB

Provides a way to interact with the program, insert dynamic jumps patches and inspect the program symbols and state.

LTTng UST

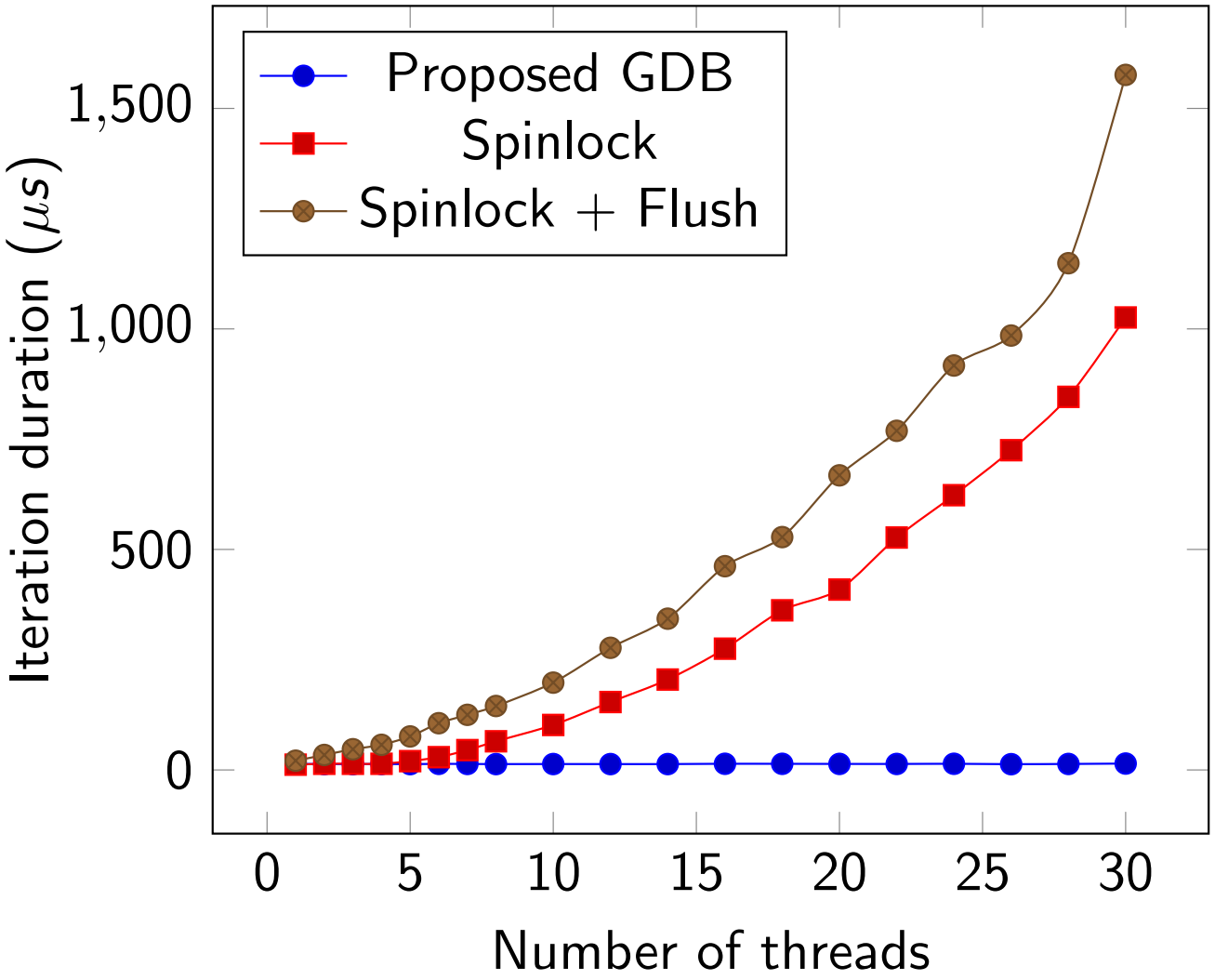
Provides a framework for fast tracepoints that scales using lockless buffers.

Implementation

GDB

- Use a library of pre-defined tracepoint functions
- Each function has a specific buffer size
- GDB select the appropriate function and links it
- Reuse most of the fast tracepoints logic

Performance comparison



The GPUOpen Initiative

GPUOpen

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

Heterogeneous System Architecture (HSA) Foundation

- Provide a standardized interface for programmer
- Multiple instruction sets
- GPUOpen is an implementation by AMD

HSA Specification

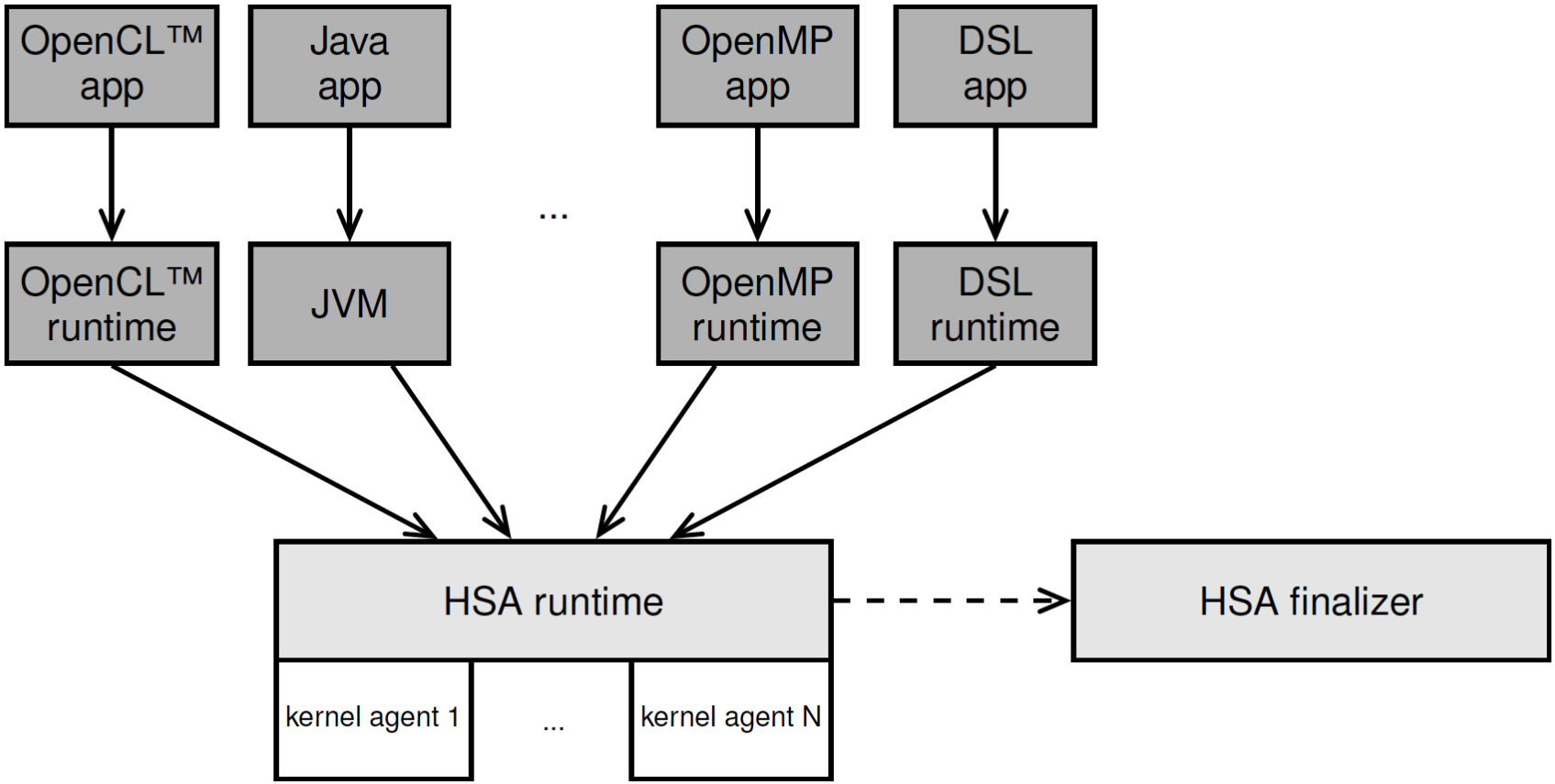


Image from HSA Foundation website

CodeXL

A tool suite developped by AMD, the version 2.0 has just been released and is now open-source. Its capacities include :

- Integrated debugging with AMD GPU
- GPU and CPU profiling
- Power profiling
- Using GPU and CPU performance counters

GDB

AMD has started to work on a debugger for their gpu using the GPUOpen software stack based on GDB 7.8.

Features

- Integrated CPU and GPU debugging
- Inspect GPU state
- Trace kernel launches

Future work

GPUOpen

The debugging and tracing challenges of heterogeneous and manycore systems will be investigated using the GPUOpen software stack.

Fiji Nano graphic card

The Fiji nano, with 4096 processors in 64 compute groups, is the main platform that will be used for these experimentations.

Any Questions ?

Contact

didier.nadeau@polymtl.ca