GDB (GNU Debugger) for AArch64 and ARM

Progress 2015 & 2016 Plans

Presented by
Yao Qi

Date
BKK16-304 March 9, 2016

Event
Linaro Connect BKK16
Overview

Syscall catchpoint for AArch64 Linux
AArch64 reverse debugging
AArch64 tracepoint
Range stepping for AArch64
AArch64 non-stop debugging
AArch64 fast tracepoint
AArch64 Multi-Arch
Support GNU vector in inferior call
Software Single Step in GDBserver for ARM

Number of GDB commits on target dependent code

Antoine Tremblay
Reverse debugging

Process record and replay

- Record the registers and memories changes by instruction
- Decode each instruction!
  - x86, aarch64, arm, moxie, and ppc

Target Vector Interface

GDB core

- wait/resume
- read/write reg/mem
- insert/remove breakpoint
- ...
Non-stop debugging

- Non-stop vs All-stop
  - All-stop: one thread stops, all threads stop
  - Non-stop: one thread stops, the rest are running,

- Pros and cons
  - Safe to step over breakpoint
  - Intrusive
Non-stop debugging (cont)

- Displaced stepping vs in-place stepping
  - Execute instruction elsewhere (on scratch pad),
  - New instructions are equivalent to the original one,
  - aarch64, arm, x86, POWER S390

Multiple threads share single scratch pad

---

[Diagram showing the process of displaced stepping vs in-place stepping with instructions and breakpoints.]
Non-stop debugging (cont)

The offset can be out of the range of imm19

GDB sets x0 to 0x400a38

How to tell the condition result?

GDB knows the condition result by PC, and adjust PC accordingly

Don't emit BL

GDB updates LR instead.
Tracepoint

- Non-intrusive for live system,
- Live analysis or saved in files for post analysis (TFILE and CTF)
- Added in GDB 7.2 by codesourcery (sponsored by Ericsson)
- Only x86 and x86_64 is supported

- Three types of tracepoint
  - Tracepoint,
    - (gdb) trace foo
  - Fast tracepoint,
    - (gdb) ftrace foo
  - Static tracepoint,
    - User-space LTTng,

![Diagram showing tracepoint creation, start tracing, stop tracing, and analyzing trace data]
Tracepoint (cont)

- Tracepoint
  - Done by breakpoint,
  - Slow

- Fast tracepoint
  - In process agent,
  - Done by jump pad,
    - Dynamically generated,
    - Save registers on stack
    - Spin lock
    - Call C function
    - Jump back
  - Doesn’t work in shared library
Multi-Arch Debugging

- AArch64 GDB can debug ARM program
  - x86_64 GDB can debug x86 program,
- Cooperate with kernel
- Handle the differences of thread area, siginfo_t, register sets and HW breakpoint/watchpoint
Plan

- Compile and inject code in GDB
  - libcc1.so in GCC 5.0 and higher,
  - X86, POWER and S390 is supported,
- Improve GDB performance in remote debugging,
- More tests to process record/reply for ARM and AArch64
- Kernel-awareness
End

Thank you