Old Code Lives Again: Tracepoints in GDB

Stan Shebs
CodeSourcery

LinuxCon Tracing Mini-Summit
August 2010

Traditional Debugging

- Debugger controls the program
- Breakpoints stop execution, step and continue resume
- Threads stop and resume together (more or less)
- Predictable behavior
- ...but maybe incorrect for real-time system

Nonstop debugging

- Debugger controls part of the program
- Breakpoints stop some threads, while others continue running
- Threads may stop and resume independently of each other
- Less predictable behavior
- ...but better for real-time system

Tracing

- Program runs freely
- Tracepoints stop program just long enough to collect data
- Minimal effect on program behavior
- ...but limited by user's foresight in collecting the right data at the right times, and by size of buffer accumulating the data

Tracing Commands

```
(gdb) trace foo.cc:45
Tracepoint 1 at 0x...
(gdb) actions 1
> collect $regs, $loc, myglob[3]@10
> end
(gdb) tstart
(gdb) tstop
(gdb) tfind 0
(gdb) print myglob[5]
$1 = 92
```

Tracing Implementation

- `tstart` command downloads tracepoint definitions including actions to target
- Agent (aka stub) installs trap instructions similarly to breakpoints, clears trace buffer
- When trap taken, agent creates a “trace frame” in buffer, adds blocks of collected data
- `tfind` command chooses a frame as source of memory and register data, agent returns from it instead of current memory/registers

Agent Expressions

- Collection of global variable needs only address and length
- Local variable needs value of base reg plus offset
- Array element needs value of array, plus index times size of element
- ...

Agent Expressions, Compiled

```
(gdb) maint agent globarr[45]
  0  const64 139928864  ; address of globarr
  9  const8 45
 11  const8 2           ; array of short
 13  mul
 14  add
 15  zero_ext 32        ; (useless?)
 17  const8 2           ; short is two bytes
 19  trace              ; collect it
 20  end
```

History of Tracing in GDB

- 1997 - Cygnus project for EMC
- 1998-1999 – Productized, shown at ESC
- 1999-2008 - umm, no customers?
- 2006 - DSLab Lanzhou implementation
- 2007 – Tracepoints for Linux kernel
- 2008-2010 - CodeSourcery project for Ericsson

The Target

- Phone switch running GNU/Linux
- Single large application with multiple threads
- Hard real-time
- Want to diagnose both in lab and in field
- No `ptrace`
- No stopping
- No hesitating

Debug Stub

- “Traditional” debug stub
  - Built into application
  - Speaks GDB remote protocol
- Runs in a dedicated thread
- Controls other threads with signals
  - `pthread_kill (SIGUSR1)`
- Collects registers from signal context

New Work

- Tracepoints become breakpoints
- Tracepoint action changes
- Non-stop with tracepoints
- Conditional tracepoints
- Trace state variables
- Fast tracepoints
- Disconnected tracing
- Trace files

Tracepoints Become Breakpoints

- Tracepoints are like breakpoints that are inserted only for tracing run and don't stop the program
- Eliminates redundant create, delete, enable, disable, etc
- Enables use of breakpoint features such as locations and conditionals
- User interface verbiage needs generalization

Tracepoint Action Changes

- More expression types
  - Comparisons (==, !=, >, >=, <, <=)
  - Logical connectives (||, &&)
  - Conditional expression (?:)
  - Assignment (trace state variables only)
  - C++ (&, this)
- DWARF location expressions for local variables
- default-collect

Non-stop with Tracepoints

- Agent must leave tracepoint traps in place, and do displaced stepping to get over them
- Similar to GDB algorithm, but purely target-side
  - Take trap
  - Single-step at displaced location
  - Fix up state (relative jumps, etc)
  - Resume at next instruction
- No changes on host side

Conditional Tracepoints

- Syntax exactly as for breakpoints
  - `(gdb) trace foo if arg == 7`
- Cuts down on uninteresting hits in hot path
- Forestalls trace buffer filling up
- Must create different kind of agent expression for conditional
  - Don't collect anything
  - Return a result

Trace State Variables

- Defined by user, managed by target
  - `tvariable $mytsv` [*initialvalue*]
  - `info tvariables`
- “Convenience variables for the target agent”
- Actions and conditionals may get and set
- New action type `teval` computes without collecting
- Predefined variable `$trace_timestamp`

Fast Tracepoints (ftrace)

- Based on jumps instead of traps
- Eliminates context switch and signal overhead
- May not be able to install anywhere (on x86, only at 5-byte-or-more insns)
- Conditional test is very fast, still needs mutex if multiple threads can hit tracepoint
- Compile bytecodes to native for even more speed

Disconnected Tracing

- Trace continues to run after detach or quit
- `set disconnected-tracing 1` to handle unexpected disconnects
- Target gets full definition including source forms
  - Upload upon reconnection, and compare
  - GDB tracepoint matches target, but numbering may be different; translate references
  - No matching GDB tracepoint; create one

Trace Files

- Disconnected target dumps trace data into a file
- `(gdb) target tfile` *filename*
- `tfind` works as with live target
- Old code refactored to use target vector
- New target “tfile”, similar to corefile target
- File format half and half
  - Textual header for protocol-like description
  - Binary body for raw trace buffer

Other Improvements

- More user feedback - `tstatus`
- Combinations
  - `tfind` while trace is running?
  - Mixed breakpoints and tracepoints
  - C++ features
    - This
    - Static members
- MI (Machine Interface) for Eclipse use

Also, GDBserver

- GDBserver now includes a tracepoint agent
- Slow tracepoints stop/continue as for breakpoints
- Libinproctrace for fast tracepoints
  - Periodic resync of in-app trace buffer with real trace buffer in GDBserver

Static Tracepoints (strace)

- Uses markers compiled into app
  - UST (User-Space Tracing library of LTTng)
- Agent uploads locations of markers
- May attach arbitrary collection actions
- $_sdata convenience variable reports compiled-in data collection

Current work

- Collecting strings
- Metadata (time started, notes, user name, ...)
- On-the-fly enable/disable
- Shorter fast tracepoints
  - 3 bytes on x86, using intermediate jump pads
- Partial data handling
  - No errors if parts of an object not collected

Availability

- GDB 7.1 -- GDB additions
- GDB 7.2 -- GDBserver additions
- Basic machinery is generic
- Fast tracepoints are arch-specific
  - Jump instruction, state save/restore required
  - Bytecode->native optional
  - Done for x86 and x86-64

Credits

- Michael Snyder (Cygnus)
- Jim Blandy (Cygnus)
- Pedro Alves (Codesourcery)
- Vladimir Prus (Codesourcery)
- Marc Khouzam (Ericsson)
- Anton Massoud (Ericsson)
- Dominique Toupin (Ericsson)