uftrace updates

Namhyung Kim <namhyung@gmail.com>

Tracing Summit 2019

uftrace

- Function (graph) tracer for userspace
  - Heavily inspired from the kernel ftrace
  - Trace single process execution flow
- Project homepage
  - https://github.com/namhyung/uftrace

Build updates

- No more mandatory dependency
  - All libraries are optional (including libelf)
  - `misc/install-deps.sh`

```
$ ./configure
uftrace detected system features:
...         prefix: /usr/local
...         libelf: [ on  ] - more flexible ELF data handling
...          libdw: [ on  ] - DWARF debug info support
...   libpython2.7: [ on  ] - python scripting support
...    libncursesw: [ on  ] - TUI support
...   cxa_demangle: [ on  ] - full demangler support with libstdc++
...     perf_event: [ on  ] - perf (PMU) event support
...       schedule: [ on  ] - scheduler event support
...       capstone: [ on  ] - full dynamic tracing support
```

Automatic arguments

- A selected set of well-known functions
  - C/Linux standard library
  - For external functions
- Parse argument spec from debug info
  - Require DWARF data in the binary
  - For internal functions

DWARF arguments

- Using libdw from elfutils
  - Optional but needed for DWARF parsing
- Primitive data types only
  - Include strings!
- Human-readable output
  - Symbolize address
  - Enum type (bitmask)

String handling

- Pointer dereference
  - Can result in a segfault
  - Make sure it’s safe before the dereference
- Current approach
  - Bulid a list of mappings (`/proc/self/maps`)
  - Deny access not in the list
  - No 100% guarantee tho

Automatic arguments

```
$ uftrace -a --force pwd
/home/namhyung/tmp
# DURATION     TID     FUNCTION
 158.460 us [141144] | getenv("POSIXLY_CORRECT") = "NULL";
   1.584 us [141144] | strrchr("pwd", '/') = "NULL";
  71.230 us [141144] | setlocale(LC_ALL, "") = "en_US.UTF-8";
   2.141 us [141144] | bindtextdomain("coreutils", "/usr/share/locale");
   1.094 us [141144] | textdomain("coreutils");
   0.845 us [141144] | __cxa_atexit();
   2.336 us [141144] | getopt_long(1, 0x7ffe173fde38, "LP") = -1;
   8.456 us [141144] | getcwd(0, 0) = "/home/namhyung/tmp";
  18.562 us [141144] | puts("/home/namhyung/tmp") = 19;
   0.977 us [141144] | free(0x557e48431520);
   1.163 us [141144] | __fpending();
   0.964 us [141144] | fileno(&_IO_2_1_stdout_) = 1;
   0.683 us [141144] | __freading();
   0.146 us [141144] | __freading();
   1.400 us [141144] | fflush(&_IO_2_1_stdout_) = 0;
   ...
```

Perf event integration

- Utilize HW and system info
  - Task creation/rename/exit/schedule
  - HW performance event counters
- Require system permission
  - `/proc/sys/kernel/perf_event_paranoid <= 2`
  - For userspace-only events

Reading perf counters

- “read” trigger
  - Work for arbitrary functions
  - Reading values at start/end of the function
  - Show the difference at the end
  - Currently general HW perf events only
    - cycles/instructions (IPC)
    - cache-reference/miss (hit ratio)
    - branch-insn/miss (predict ratio)

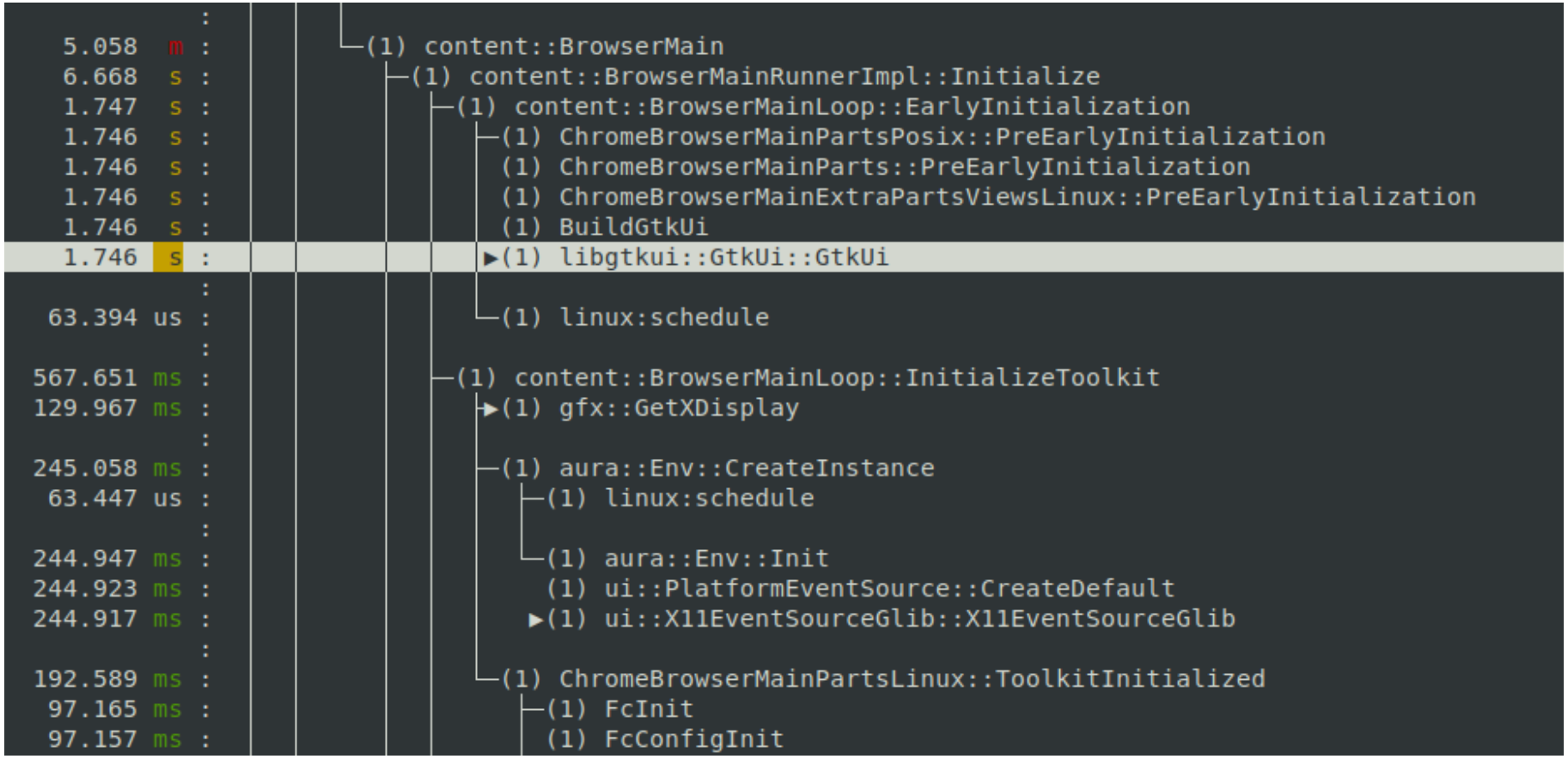
Reading perf counters

```
$ uftrace -T foo@read=pmu-cycle foobar
# DURATION     TID     FUNCTION
            [143355] | main() {
   1.454 us [143355] |   calloc();
            [143355] |   foo() {
            [143355] |     /* read:pmu-cycle (cycle=618,
                     |              instructions=27) */
   0.239 us [143355] |     bar();
   0.130 us [143355] |     bar();
            [143355] |     /* diff:pmu-cycle (cycle=+3683,
                     |              instructions=+2626, IPC=0.71) */
   9.588 us [143355] |   } /* foo */
   1.226 us [143355] |   free();
  18.582 us [143355] | } /* main */
```

TUI (Text User Interface)

- Using libncursesw
- Support analysis commands
  - `report`, `graph`, `info`
  - Source line information
- Planned
  - `replay`
  - Dynamic filter change

uftrace tui


5.058 m : (1) content::BrowserMain
6.668 s : (1) content::BrowserMainRunnerImpl::Initialize
1.747 s : (1) content::BrowserMainLoop::EarlyInitialization
1.746 s : (1) ChromeBrowserMainPartsPosix::PreEarlyInitialization
1.746 s : (1) ChromeBrowserMainParts::PreEarlyInitialization
1.746 s : (1) ChromeBrowserMainExtraPartsViewsLinux::PreEarlyInitialization
1.746 s : (1) BuildGtkUi
1.746 s : ▶(1) libgtkui::GtkUi::GtkUi
63.394 us : (1) linux:schedule
567.651 ms : (1) content::BrowserMainLoop::InitializeToolkit
129.967 ms : ▶(1) gfx::GetXDisplay
245.058 ms : (1) aura::Env::CreateInstance
63.447 us : (1) linux:schedule
244.947 ms : (1) aura::Env::Init
244.923 ms : (1) ui::PlatformEventSource::CreateDefault
244.917 ms : ▶(1) ui::X11EventSourceGlib::X11EventSourceGlib
192.589 ms : (1) ChromeBrowserMainPartsLinux::ToolkitInitialized
97.165 ms : (1) FcInit
97.157 ms : (1) FcConfigInit

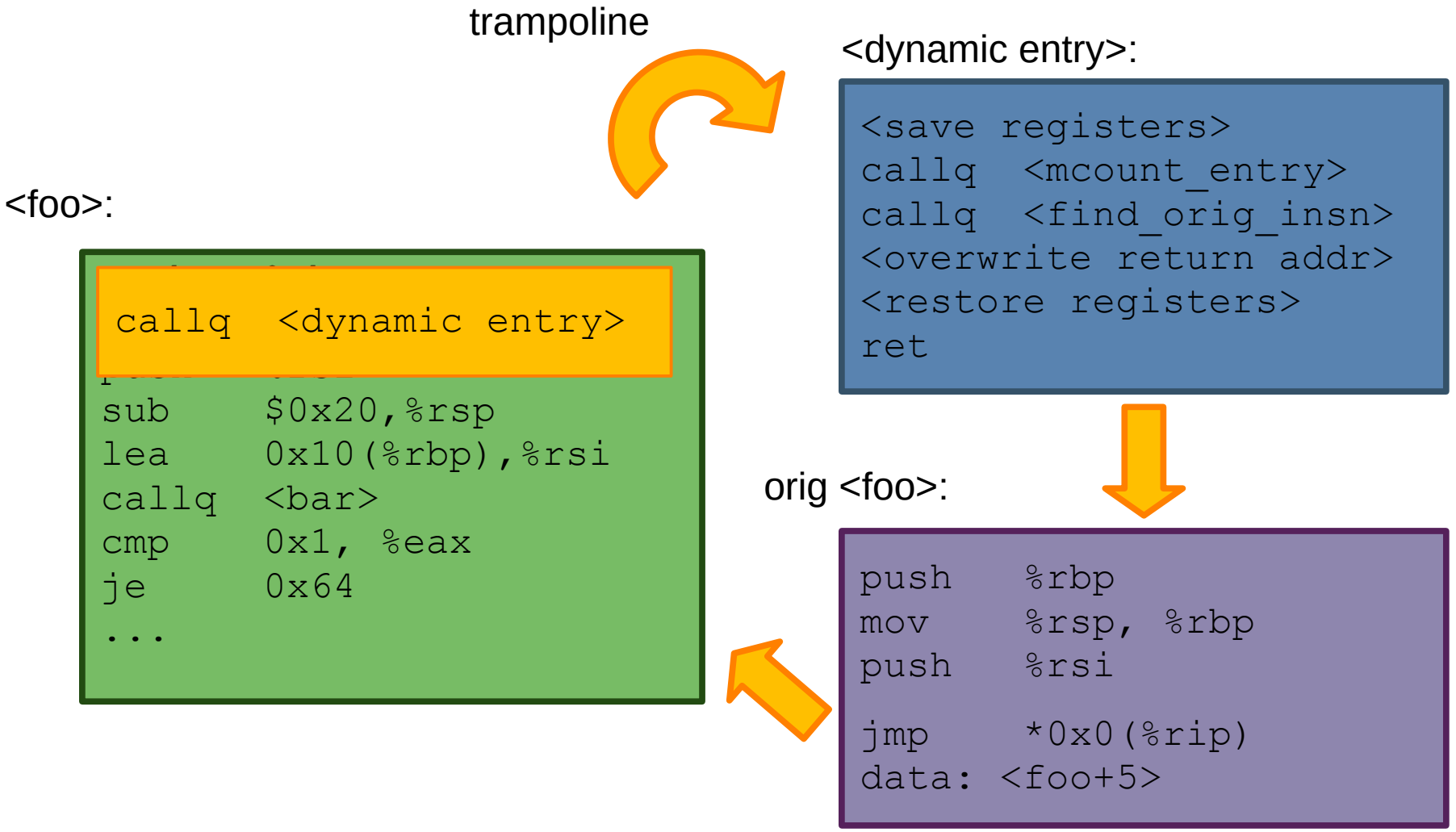
Dynamic tracing

- Basic operation
  - Compiler adds NOP in the prologue
  - Selectively enable functions
- Problems
  - Compiler support
    - Recompilation
  - PIC/PIE not working with `-mnop-mcount`

Dynamic tracing update

- Full dynamic tracing
  - Without compiler support
    - Need symbol table in the binary
    - Work on x86 (experimental)
- How?
  - Parse instructions using libcapstone
  - Save original instructions somewhere
  - Overwrite original with call to mcount

Dynamic tracing



Instruction parsing

- Simple approach
  - Avoid PC-relative addressing
  - Check operands of instruction
  - No (back) jump to the prologue
  - No jump to the middle of function
  - No indirect jump

Success rate

```
$ uftrace -P . -v  uftrace replay
...
dynamic: dynamic patch type: 0 (none)
dynamic: dynamic patch stats for 'uftrace'
dynamic:    total:      752
dynamic:  patched:      558 (74.20%)
dynamic:   failed:      185 (24.60%)
dynamic:  skipped:        9 ( 1.19%)
dynamic: no match:        0
...

# DURATION     TID     FUNCTION
            [ 54615] | argp_parse() {
   0.981 us [ 54615] |   parse_option();
   0.533 us [ 54615] |   parse_option();
   ...
```

Another approach

- Dynamic unpatching
  - Not deal with instruction parsing
    - Leave the compiler do the job
  - Ignore non-interested functions
    - By changing them to NOP
  - Work on x86 (in review)

Size filter

- Enable/disable dynamic tracing
  - Based on the size of function
  - To reduce amount of trace data
  - Assume small sized function is not important?

Register saving

- Caller saved
  - Scratch registers, argument passing
  - Can be changed during function call
- Callee saved
  - Guaranteed to keep same value after function call

Compiler strikes back

- Gcc has `-fipa-ra`
  - Use scratch registers across functions
  - Only if it can see its definition

```
-fipa-ra
    Use caller save registers for allocation
    if those registers are not used by any called function.
    In that case it is not necessary to save and restore
    them around calls.  This is only possible if
    called functions are part of same compilation unit
    as current function and they are compiled before it.

    Enabled at levels -O2, -O3, -Os.
```

Return address handling

- Difficulties dealing with it
  - uftrace wants to trace function returns
  - It changes return address in the stack
  - Some applications assume some condition of return address
  - System libraries unwind frames internally
    - C++ exception
    - backtrace
    - Non-local goto

Recovering return address

- Compiler support
  - `gcc -finstrument-functions`
  - `clang -fxray-instrument`
- Recover trigger
  - Restore original return address
  - `uftrace -T foo@recover`
- Auto-recover
  - Recover parent when entering to child

Q & A

- Thanks!

https://github.com/namhyung/uftrace