Implementation of Dynamic Patching/Unpatching in `uftrace`

Gabriel-Andrew Pollo-Guilbert
gabriel.pollo-guilbert@polymtl.ca

June 1, 2021

Quick primer on uftrace

uftrace is a command-line tool for quickly tracing function calls in a program.

```c
#include <stdio.h>

void a() {
    puts("a() called\n");
}

void b() {
    puts("b() called\n");
    a();
}

void c() {
    puts("c() called\n");
    b();
}

int main() {
    a();
    putchar('\n');
    b();
    putchar('\n');
    c();

    return 0;
}
```

```
# DURATION     TID     FUNCTION
   1.834 us [598864] | __monstartup();
   1.242 us [598864] | __cxa_atexit();
            [598864] | main() {
            [598864] |   a() {
  13.456 us [598864] |     puts();
  14.327 us [598864] |   } /* a */
   2.334 us [598864] |   putchar();
            [598864] |   b() {
   1.503 us [598864] |     puts();
            [598864] |     a() {
   1.263 us [598864] |       puts();
   1.763 us [598864] |     } /* a */
   4.138 us [598864] |   } /* b */
   0.922 us [598864] |   putchar();
            [598864] |   c() {
   1.182 us [598864] |     puts();
            [598864] |     b() {
   1.132 us [598864] |       puts();
            [598864] |       a() {
   1.082 us [598864] |         puts();
   1.553 us [598864] |       } /* a */
   3.316 us [598864] |     } /* b */
   5.149 us [598864] |   } /* c */
  28.745 us [598864] | } /* main */
```

Compiler static tracepoints support in uftrace

```
$ gcc main.c

push   %rbp
mov    %rsp,%rbp

mov    $0x0,%edi
call   1119
pop    %rbp
ret
```

```
$ gcc main.c -pg

push   %rbp
mov    %rsp,%rbp
call   mcount@GLIBC_2.2.5
mov    $0x0,%edi
call   11a9
pop    %rbp
ret
```

```
$ gcc main.c -pg -mfentry

call   __fentry__@GLIBC_2.13
push   %rbp
mov    %rsp,%rbp

mov    $0x0,%edi
call   11a9
pop    %rbp
ret
```

```
$ gcc main.c -finstrument-functions
push   %rbp
mov    %rsp,%rbp
push   %rbx
sub    $0x8,%rsp
mov    0x8(%rbp),%rax
mov    %rax,%rsi
lea    -0x17(%rip),%rdi
call   __cyg_profile_func_enter@plt
mov    $0x0,%edi
call   1149
mov    %eax,%ebx
mov    0x8(%rbp),%rax
mov    %rax,%rsi
lea    -0x36(%rip),%rdi
call   __cyg_profile_func_exit@plt
mov    %ebx,%eax
mov    -0x8(%rbp),%rbx
pop    %rbp
ret
```

Compiler dynamic tracepoints support in `uftrace`

```
$ gcc main.c
```

```
push   %rbp
mov    %rsp,%rbp
mov    $0x0,%edi
call   1119
pop    %rbp
ret
```

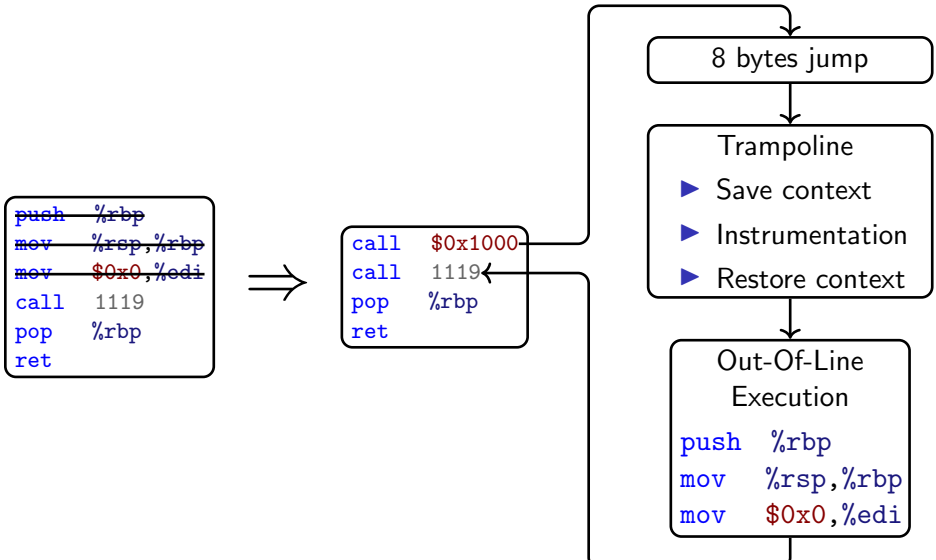
```
$ gcc main.c -pg -mfentry -mnop-mcount
```

```
nop                # 5 bytes
push   %rbp
mov    %rsp,%rbp
mov    $0x0,%edi
call   11a9
pop    %rbp
ret
```

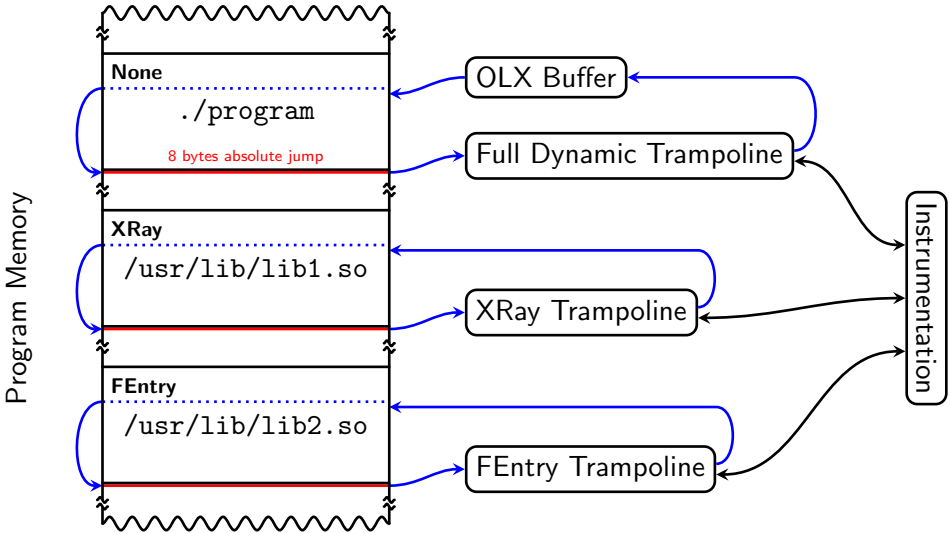
```
$ clang main.c -fxray-instrument -fxray-instruction-threshold=1
```

```
        jmp    prologue   # 2 bytes
        nop               # 9 bytes
prologue: push %rbp
        mov    %rsp,%rbp
        mov    $0x0,%edi
        call   11a9
        pop    %rbp
        ret                # 1 byte
        nop                # 10 bytes
```

Full dynamic tracepoints support in `uftrace`



Overview of dynamic tracepoints in `uftrace`



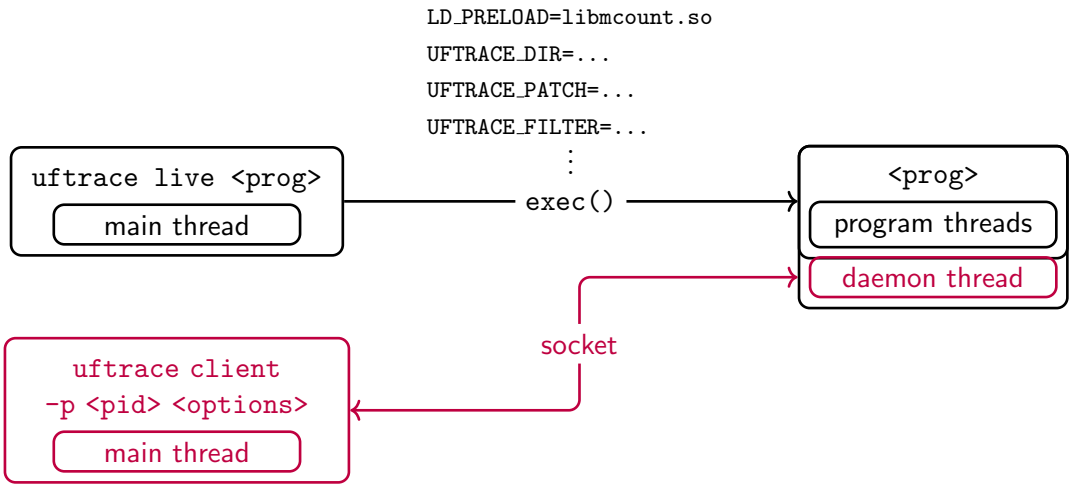
The missing features in `uftrace`

- The selected tracepoints are only enabled/disabled at program startup.
- You forgot to enable a tracepoint? Too bad, start the program over.
- In fact, you can't change any `uftrace` parameters after startup.

Ongoing work for `uftrace` by DORSAL

- Implementation of a command daemon inside the traced application. (Clément Guidi)
- Support for enabling/disabling x86 tracepoints concurrently. (Gabriel-Andrew Pollo-Guilbert)
- Basic support for enabling/disabling ARM tracepoints. (Misha Krieger-Raynauld)

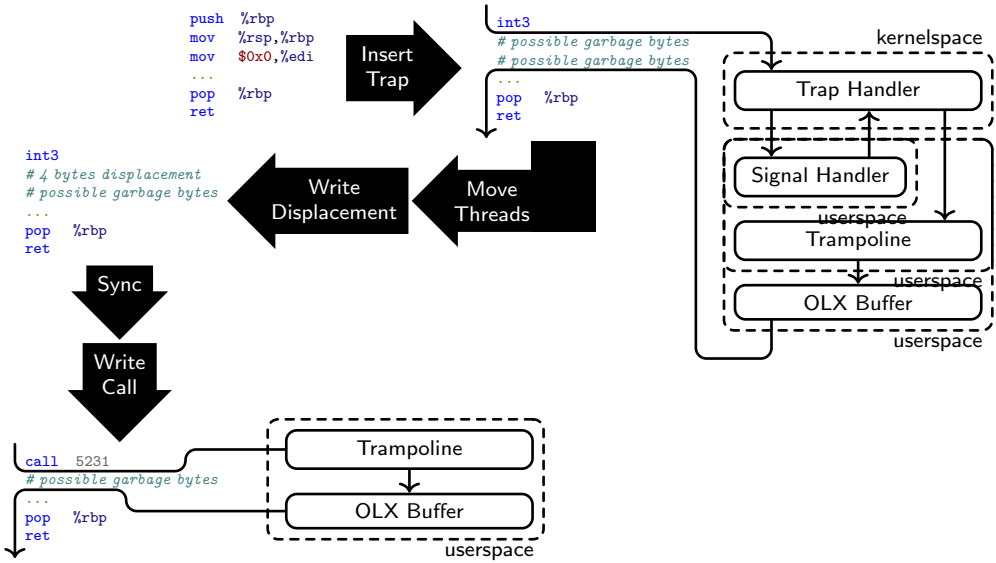
uftrace architecture with the command daemon



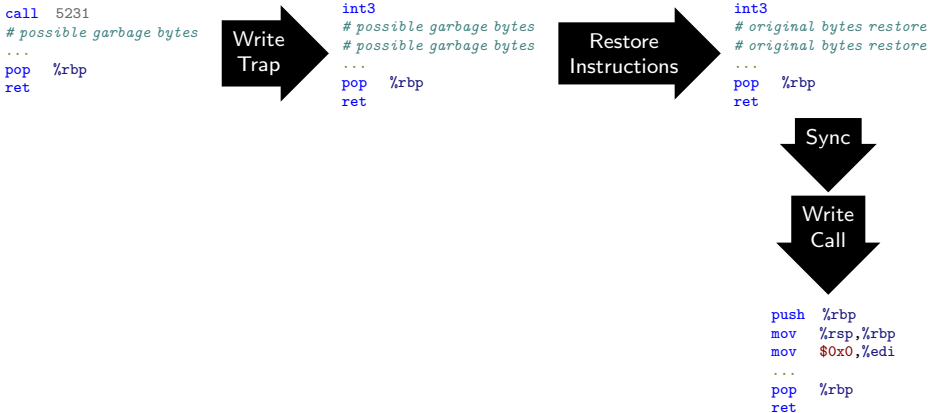
The challenges of fully dynamic tracepoints

- Safety checks to ensure patching won't break the program.
- Generating OLX buffers and fixing PC-relative instructions.
- Special care must be used when overwriting instructions concurrently.
- The modifications may straddle multiple cache lines.
- And more...

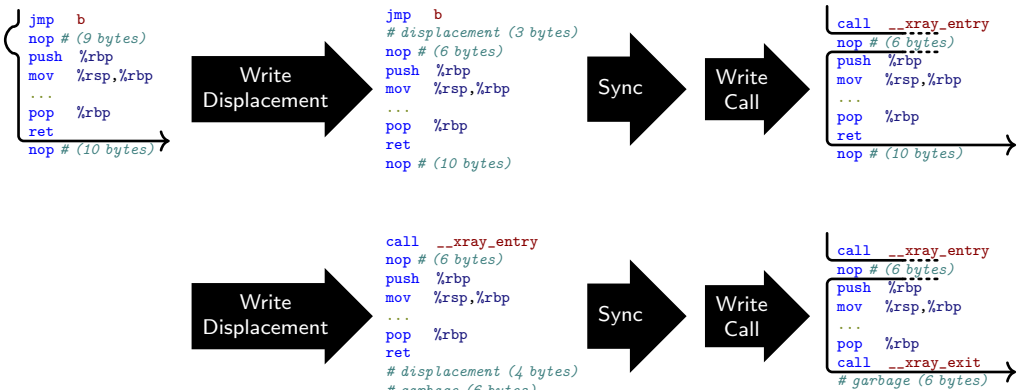
Fully dynamic tracepoing patching sequence



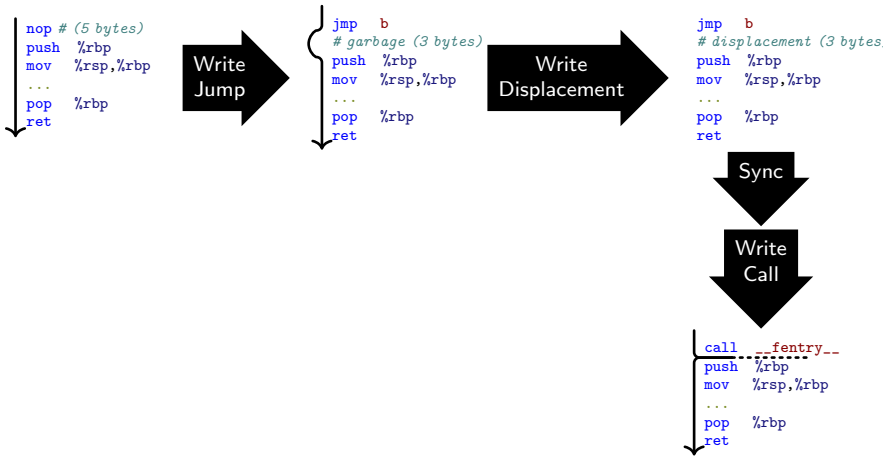
Fully dynamic tracepoing unpatching sequence



XRay tracepoing patching sequence



FEntry tracepoing patching sequence



Demo / Questions?