TRYING TO USE UPROBES AND BPF ON NON-C USERSPACE

Arnaldo Carvalho de Melo
acme@redhat.com

WHAT IS THIS ABOUT?

- User space
- uprobes
- !C
- Calling conventions
- A report from the field

!C

- Go, Rust, Zig
- Looked mostly at go so far
- Try and keep same workflow as for other languages
- Improving support on the observability toolchest
- There are new tools, use it...
- ... but using familiar tools helps

WHO ASKED FOR THIS?

- Red Hat Customer
- Telco
- Lots of software providers
- Wanting more metrics
- Took place a few months ago

NATIVE METRICS

- prometheus and others
- Existing metrics in the observed software
- But I need some more!
- Convince the sofware authors to add natively
- Wait for next version?

UPROBES

- Collect some more metrics
- More flexibility on using existing ones
- Meta metrics using BPF maps
- Performance degradation in hot metrics
- Next version can come with these new ones

UPROBES 2

- Binaries have lots of info
- DWARF
- Coding conventions
- Tooling to query this

DWARF

- pahole
- types
- functions
- perf

BUT...

- What about golang, Rust, zig?
- Compiled
- Have DWARF
- Calling conventions

GOLANG

- Language used in the first telco sw provider
- prometheus used
- Lets attempt to use uprobes on its API

GO LOOKING

- DWARF produced had some issues
- Fixed pahole
- 'perf probe' worked

PAHOLE

- API hunt
- pfunct looking for methods
- pahole looking for structs

METHODS WITH PAHOLE

```
# pfunct --prototype tests/prometheus/main | grep -w counter
void vendor/golang.org/x/crypto/chacha20.(*Cipher).SetCounter(vendor/golang.org/x/crypto/chacha20.Cipher *s, ui
void crypto/cipher.(*gcm).deriveCounter(crypto/cipher.gcm *g, uint8 *counter, struct []uint8 nonce);
void crypto/cipher.(*gcm).counterCrypt(crypto/cipher.gcm *g, struct []uint8 out, struct []uint8 in, uint8 *coun
void github.com/prometheus/client_golang/prometheus.(*counter).Describe(chan<- *github.com/prometheus/client_go
void github.com/prometheus/client_golang/prometheus.(*counter).Collect(chan<- github.com/prometheus/client_gola
void github.com/prometheus/client_golang/prometheus.(*counter).Inc(github.com/prometheus/client_golang/promethe
void github.com/prometheus/client_golang/prometheus.(*counter).get(github.com/prometheus/client_golang/promethe
void github.com/prometheus/client_golang/prometheus.(*counter).Add(github.com/prometheus/client_golang/promethe
#
```

METHODS WITH PERF PROBE

```
# perf probe -x tests/prometheus/main -F *Inc*
crypto/x509.IncorrectPasswordError
github.com/prometheus/client_golang/prometheus.(*counter).Inc
github.com/prometheus/client_golang/prometheus.(*gauge).Inc
github.com/prometheus/client_golang/prometheus.errInconsistentCardi
github.com/prometheus/common/expfmt.NegotiateIncludingOpenMetrics
vendor/golang.org/x/crypto/chacha20poly1305.avx2IncMask
vendor/golang.org/x/crypto/chacha20poly1305.sseIncMask
vendor/golang.org/x/text/transform.errInconsistentByteCount
#
```

GO STRUCTS

```
# pahole tests/prometheus/main -C github.com/prometheus/client_golang/prometheus.counter
struct github.com/prometheus/client_golang/prometheus.counter {
        uint64                     valBits;                                      /*   0   8 */
        uint64                     valInt;                                       /*   8   8 */
        github.com/prometheus/client_golang/prometheus.selfCollector selfCollector; /*  16  16 */
        github.com/prometheus/client_golang/prometheus.Desc * desc;              /*  32   8 */
        struct []*github.com/prometheus/client_model/go.LabelPair labelPairs;    /*  40  24 */
        /* --- cacheline 1 boundary (64 bytes) --- */
        sync/atomic.Value          exemplar;                                     /*  64  16 */
        func() time.Time           now;                                          /*  80   8 */

        /* size: 88, cachelines: 2, members: 7 */
        /* last cacheline: 24 bytes */
};

#
```

PERF PROBE FOR C

- List function source code?
- Look for what variables can be collected

C, KERNEL

```
# perf probe -L kfree
```

C, KERNEL

```
# perf probe -L kfree
```

```
<kfree@/usr/src/debug/kernel-6.4.10/linux-6.4.10-200.fc38.x86_64/mm/slab_common.c:0>
      0  void kfree(const void *object)
         {
      2          struct folio *folio;
                 struct slab *slab;
                 struct kmem_cache *s;

                 trace_kfree(_RET_IP_, object);

      8          if (unlikely(ZERO_OR_NULL_PTR(object)))
                         return;

     11          folio = virt_to_folio(object);
     12          if (unlikely(!folio_test_slab(folio))) {
     13                  free_large_kmalloc(folio, (void *)object);
                         return;
                 }

     17          slab = folio_slab(folio);
                 s = slab->slab_cache;
     19          __kmem_cache_free(s, (void *)object, _RET_IP_);
         }
         EXPORT_SYMBOL(kfree);
#
```

C, USERSPACE

```
# perf probe -x prometheusnoop -L prometheusnoop_bpf__open_opts
```

C, USERSPACE

```
# perf probe -x prometheusnoop -L prometheusnoop_bpf__open_opts


      0  prometheusnoop_bpf__open_opts(const struct bpf_object_open_opts *opts)
         {
                struct prometheusnoop_bpf *obj;
                int err;

      5         obj = (struct prometheusnoop_bpf *)calloc(1, sizeof(*obj));
      6         if (!obj) {
      7                 errno = ENOMEM;
      8                 return NULL;
                }

     11         err = prometheusnoop_bpf__create_skeleton(obj);
     12         if (err)
     13                 goto err_out;

     15         err = bpf_object__open_skeleton(obj->skeleton, opts);
     16         if (err)
     17                 goto err_out;

     19         return obj;
         err_out:
     21         prometheusnoop_bpf__destroy(obj);
     22         errno = -err;
     23         return NULL;
     24  }

         static inline struct prometheusnoop_bpf *
         prometheusnoop_bpf__open(void)

#
```

VARIABLES

```
# perf probe -x /lib64/libc.so.6 -V malloc
```

VARIABLES

```
# perf probe -x /lib64/libc.so.6 -V malloc

Available variables at malloc
        @<__libc_malloc+0>
                char*   __PRETTY_FUNCTION__
                size_t  bytes
#
```

VARIABLES

```
# perf probe -x /lib64/libc.so.6 -V malloc

Available variables at malloc
        @<__libc_malloc+0>
                char*   __PRETTY_FUNCTION__
                size_t  bytes
#


# perf probe -x /lib64/libc.so.6 __libc_malloc bytes
Added new event:
  probe_libc:__libc_malloc (on __libc_malloc in /usr/lib64/libc.so.

You can now use it in all perf tools, such as:

        perf record -e probe_libc:__libc_malloc -aR sleep 1
#
```

USE IT

```
# perf trace -e probe_libc:__libc_malloc/max-stack=8/ --max-events=2
```

USE IT

```
# perf trace -e probe_libc:__libc_malloc/max-stack=8/ --max-events=2

  0.000 gnome-control-/405742 probe_libc:__libc_malloc(__probe_ip: 139765834955664, bytes: 16)
                                malloc (/usr/lib64/libc.so.6)
                                g_slice_alloc (/usr/lib64/libglib-2.0.so.0.7600.5)
                                g_object_notify_queue_freeze.lto_priv.0 (/usr/lib64/libgobject-2.0.so.0.7600.
                                g_object_freeze_notify (/usr/lib64/libgobject-2.0.so.0.7600.5)
                                gtk_window_set_default_size_internal (/usr/lib64/libgtk-4.so.1.1000.5)
                                toplevel_compute_size (/usr/lib64/libgtk-4.so.1.1000.5)
                                g_closure_invoke (/usr/lib64/libgobject-2.0.so.0.7600.5)
                                signal_emit_unlocked_R.isra.0 (/usr/lib64/libgobject-2.0.so.0.7600.5)
```

USE IT

```
# perf trace -e probe_libc:__libc_malloc/max-stack=8/ --max-events=2

  0.000 gnome-control-/405742 probe_libc:__libc_malloc(__probe_ip: 139765834955664, bytes: 16)
                                malloc (/usr/lib64/libc.so.6)
                                g_slice_alloc (/usr/lib64/libglib-2.0.so.0.7600.5)
                                g_object_notify_queue_freeze.lto_priv.0 (/usr/lib64/libgobject-2.0.so.0.7600.
                                g_object_freeze_notify (/usr/lib64/libgobject-2.0.so.0.7600.5)
                                gtk_window_set_default_size_internal (/usr/lib64/libgtk-4.so.1.1000.5)
                                toplevel_compute_size (/usr/lib64/libgtk-4.so.1.1000.5)
                                g_closure_invoke (/usr/lib64/libgobject-2.0.so.0.7600.5)
                                signal_emit_unlocked_R.isra.0 (/usr/lib64/libgobject-2.0.so.0.7600.5)

  0.014 gnome-control-/405742 probe_libc:__libc_malloc(__probe_ip: 139765834955664, bytes: 48)
                                malloc (/usr/lib64/libc.so.6)
                                g_slice_alloc (/usr/lib64/libglib-2.0.so.0.7600.5)
                                g_source_new (/usr/lib64/libglib-2.0.so.0.7600.5)
                                timeout_add_full.constprop.0 (/usr/lib64/libglib-2.0.so.0.7600.5)
                                maybe_start_idle (/usr/lib64/libgtk-4.so.1.1000.5)
                                gdk_frame_clock_paint_idle (/usr/lib64/libgtk-4.so.1.1000.5)
                                g_timeout_dispatch (/usr/lib64/libglib-2.0.so.0.7600.5)
                                g_main_context_dispatch (/usr/lib64/libglib-2.0.so.0.7600.5)
#
```

PERF PROBE FOR GO

- List function source code?
- Look for what variables can be collected

GO SOURCE CODE?

```
# perf probe -x tests/prometheus/main -L 'github.com/prometheus/client_golang/prometheus.(*counter).Inc'
Debuginfo analysis failed.
  Error: Failed to show lines.
#
```

BUT ADDS A PROBE!

```
# perf probe -x tests/prometheus/main \
        'counter_inc=github.com/prometheus/client_golang/prometheus.(*counter).Inc'
Added new event:
  probe_main:counter_inc (on github.com/prometheus/client_golang/prometheus.(*counter).Inc in
                          /home/acme/git/libbpf-bootstrap/examples/c/tests/prometheus/main)

You can now use it in all perf tools, such as:

        perf record -e probe_main:counter_inc -aR sleep 1

#
```

BUT ADDS A PROBE!

```
# perf probe -x tests/prometheus/main \
        'counter_inc=github.com/prometheus/client_golang/prometheus.(*counter).Inc'
Added new event:
  probe_main:counter_inc (on github.com/prometheus/client_golang/prometheus.(*counter).Inc in
                          /home/acme/git/libbpf-bootstrap/examples/c/tests/prometheus/main)

You can now use it in all perf tools, such as:

        perf record -e probe_main:counter_inc -aR sleep 1

#
```

```
# perf probe -l
  probe_main:counter_inc (on github.com/prometheus/client_golang/prometheus.(*counter).Inc in
                          /home/acme/git/libbpf-bootstrap/examples/c/tests/prometheus/main)
#
```

USING IT

```
# perf trace -e probe_main:counter_inc/max-stack=8/ --max-events=2
  0.000 main/502040 probe_main:counter_inc(__probe_ip: 8072576)
                                  github.com/prometheus/client_golang/prometheus.(*counter).Inc
                                                       (/home/acme/git/libbpf-bootstrap/examples/c/tests/prometheus/ma
                                  runtime.goexit.abi0
                                                       (/home/acme/git/libbpf-bootstrap/examples/c/tests/prometheus/ma
  0.033 main/502040 probe_main:counter_inc(__probe_ip: 8072576)
                                  github.com/prometheus/client_golang/prometheus.(*counter).Inc
                                                       (/home/acme/git/libbpf-bootstrap/examples/c/tests/prometheus/ma
                                  runtime.goexit.abi0
                                                       (/home/acme/git/libbpf-bootstrap/examples/c/tests/prometheus/ma
#
```

MIXING WITH OTHER EVENTS

- To correlate events in non-C userspace
- With the kernel
- Other userspace components

EXAMPLE

```
# perf trace -e probe_main:counter_inc,connect
   0.000 (0.064 ms): DNS Resolver #/7247 connect(fd: 94, uservaddr: { family: LOCAL,
                                                     path: /run/systemd/resolve/io.systemd.Resolve },
                                             addrlen: 42) = 0
   2.344 (0.058 ms): DNS Resolver #/7247 connect(fd: 94, uservaddr: { family: INET6, port: 0,
                                                     addr: 2800:3f0:4004:809::200e },
                                             addrlen: 28) = -1 ENETUNREACH (Network is unreachable)
   2.418 (0.012 ms): DNS Resolver #/7247 connect(fd: 94, uservaddr: { family: UNSPEC }, addrlen: 16) = 0
   2.443 (0.029 ms): DNS Resolver #/7247 connect(fd: 94, uservaddr: { family: INET, port: 0,
                                                     addr: 142.251.135.142 }, addrlen: 16) = 0
   9.020 (0.057 ms): DNS Resolver #/7254 connect(fd: 94, uservaddr: { family: LOCAL,
                                                     path: /run/systemd/resolve/io.systemd.Resolve },
                                             addrlen: 42) = 0
  11.064 (0.044 ms): DNS Resolver #/7254 connect(fd: 94, uservaddr: { family: INET6, port: 0,
                                                     addr: 2800:3f0:4004:809::200e },
                                             addrlen: 28) = -1 ENETUNREACH (Network is unreachable)
  11.121 (0.011 ms): DNS Resolver #/7254 connect(fd: 94, uservaddr: { family: UNSPEC }, addrlen: 16) = 0
  11.141 (0.019 ms): DNS Resolver #/7254 connect(fd: 94, uservaddr: { family: INET, port: 0,
                                                     addr: 142.251.135.142 },
                                             addrlen: 16) = 0
 485.958 (        ): main/502042 probe_main:counter_inc(__probe_ip: 8072576)
 486.006 (        ): main/502042 probe_main:counter_inc(__probe_ip: 8072576)
1485.959 (        ): main/502042 probe_main:counter_inc(__probe_ip: 8072576)
^C#
```

BPFTRACE

```
# bpftrace -e 'uprobe:tests/prometheus/main:github.com/prometheus/client_golang/prometheus.(*counter).Inc { pri
stdin:1:1-78: ERROR: syntax error, unexpected (, expecting {
uprobe:tests/prometheus/main:github.com/prometheus/client_golang/prometheus.(*counter).Inc { printf("in here\n"
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

BPFTRACE

```
# bpftrace -e 'uprobe:tests/prometheus/main:github.com/prometheus/client_golang/prometheus.(*counter).Inc { pri
stdin:1:1-78: ERROR: syntax error, unexpected (, expecting {
uprobe:tests/prometheus/main:github.com/prometheus/client_golang/prometheus.(*counter).Inc { printf("in here\n"
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# bpftrace -e 'uprobe:tests/prometheus/main:github.com/prometheus/client_golang/prometheus*Inc { printf("in here
Attaching 2 probes...
in here
in here
in here
in here
in here
in here
^C

#
```

PROMETHEUSNOOP

- libbpf bootstrap
- uprobes
- uretprobes
- limitations...

GO CALLING CONVENTION

- floating point registers
- we can collect them in structs
- not in function arguments
- uprobes gets what is in struct pt_regs
- kernel doesn't touch xmm registers

PROMETHEUS EXAMPLE

```
var fake_counter = prometheus.NewCounter(prometheus.CounterOpts{
        Name: "fake_counter",
        Help: "Increments at every second",
})
```

PROMETHEUS EXAMPLE

```
var fake_counter = prometheus.NewCounter(prometheus.CounterOpts{
        Name: "fake_counter",
        Help: "Increments at every second",
})
```

```
        go func() {
                for {
                        select {
                        case <-ticker.C:
                                fake_counter.Inc()
                        }
                }
        }()
```

START THE GUINEA PIG

```
$ tests/prometheus/main
Prometheus demo
I0908 12:21:38.448655  495592 main.go:64] Starting metrics server a
```

RUNNING IT

```
^C# ./prometheusnoop --include_description --binary tests/prometheus/main
TIME     EVENT(Object)             PID
12:21:39 (0xc00021e3c0) 495592 : desc: "another_fake_gauge" value: 0.000000
12:21:39 (0xc00021e440) 495592 : desc: "sub_fake_gauge" value: 0.000000
12:21:39 (0xc00021e400) 495592 : desc: "dec_fake_gauge" value: 0.000000
12:21:40 (0xc0002000c0) 495592 : desc: "fake_counter" value: 0
12:21:41 (0xc000200120) 495592 : desc: "another_fake_counter" value: 0
12:21:41 (0xc000200120) 495592 : desc: "another_fake_counter" value: 1
12:21:42 (0xc00021e380) 495592 : desc: "fake_gauge" value: 0.000000
12:21:43 (0xc00021e3c0) 495592 : desc: "another_fake_gauge" value: 5.000000
12:21:43 (0xc00021e440) 495592 : desc: "sub_fake_gauge" value: -7.000000
12:21:43 (0xc00021e400) 495592 : desc: "dec_fake_gauge" value: -1.000000
12:21:44 (0xc0002000c0) 495592 : desc: "fake_counter" value: 1
^C#
```

READING GO STRUCTS

- DWARF
- DW_TAG_subroutine_type with DW_AT_byte_size
- DW_TAG_constant
- Both first seen in go
- Supported in pahole > 1.25

A GO STRUCT

```
# pahole -C github.com/prometheus/client_golang/prometheus.counter tests/prometheus/main
struct github.com/prometheus/client_golang/prometheus.counter {
    uint64                     valBits;                                  /*   0   8 */
    uint64                     valInt;                                   /*   8   8 */
    github.com/prometheus/client_golang/prometheus.selfCollector selfCollector; /*  16  16 */
    github.com/prometheus/client_golang/prometheus.Desc * desc;          /*  32   8 */
    struct []*github.com/prometheus/client_model/go.LabelPair labelPairs; /*  40  24 */
    /* --- cacheline 1 boundary (64 bytes) --- */
    sync/atomic.Value          exemplar;                                 /*  64  16 */
    func() time.Time           now;                                      /*  80   8 */

    /* size: 88, cachelines: 2, members: 7 */
    /* last cacheline: 24 bytes */
};
```

THE DESCRIPTION

```
# pahole -C github.com/prometheus/client_golang/prometheus.Desc tests/prometheus/main
struct github.com/prometheus/client_golang/prometheus.Desc {
    struct string              fqName;                                   /*   0  16 */
    struct string              help;                                     /*  16  16 */
    struct []*github.com/prometheus/client_model/go.LabelPair constLabelPairs; /*  32  24 */
    struct []string            variableLabels;                           /*  56  24 */
    /* --- cacheline 1 boundary (64 bytes) was 16 bytes ago --- */
    uint64                     id;                                       /*  80   8 */
    uint64                     dimHash;                                  /*  88   8 */
    error                      err;                                      /*  96  16 */

    /* size: 112, cachelines: 2, members: 7 */
    /* last cacheline: 48 bytes */
};
```

FINALLY

```
# pahole -C string tests/prometheus/main
struct string {
    uint8 *                  str;          /*     0     8 */
    int                      len;          /*     8     8 */

    /* size: 16, cachelines: 1, members: 2 */
    /* last cacheline: 16 bytes */
};
```

LIBBPF SKEL TOOL IN C

- Craft C types from pahole output
- Manual, could be automated
- bpf_probe_read_user the fields
- Object is in ctx->ax
- Documented at github

EXAMPLE

```
$ objdump -S --disassemble='github.com/prometheus/client_golang/prometheus.(*gauge).Add' \
        tests/prometheus/main | head -30
tests/prometheus/main:     file format elf64-x86-64

000000000079dec0 :
func (g *gauge) Dec() {
        g.Add(-1)
}

func (g *gauge) Add(val float64) {
        for {
  79dec0:       eb 03                jmp    79dec5 <github.com/prometheus/client_golang/prometheus.(*gauge).Add+
                oldBits := atomic.LoadUint64(&g.valBits)
  79dec2:       48 89 d8             mov    %rbx,%rax
  79dec5:       48 8b 08             mov    (%rax),%rcx

// Float64frombits returns the floating-point number corresponding
// to the IEEE 754 binary representation b, with the sign bit of b
// and the result in the same bit position.
// Float64frombits(Float64bits(x)) == x.
func Float64frombits(b uint64) float64 { return *(*float64)(unsafe.Pointer(&b)) }
  79dec8:       66 48 0f 6e c9       movq   %rcx,%xmm1
                newBits := math.Float64bits(math.Float64frombits(oldBits) + val)
  79decd:       f2 0f 58 c8          addsd  %xmm0,%xmm1
func Float64bits(f float64) uint64 { return *(*uint64)(unsafe.Pointer(&f)) }
  79ded1:       66 48 0f 7e ca       movq   %xmm1,%rdx
func (g *gauge) Add(val float64) {
  79ded6:       48 89 c3             mov    %rax,%rbx
$
```

ADD METHOD

- Uses the AVX register %xmm0 to pass the increment
- probes nor uprobes can't access those registers
- New bpf_register_read() helper
- David Marchevsky submitted an attempt at that

THE MINIMAL HELPER

```
BPF_CALL_1(bpf_read_64bit_xmm_register, u32, regindex)
{
	u64 ret;
	asm volatile("movq %%xmm0, %0" : "=r" (ret));
	return ret;
}
```

THE MINIMAL HELPER

```
BPF_CALL_1(bpf_read_64bit_xmm_register, u32, regindex)
{
	u64 ret;
	asm volatile("movq %%xmm0, %0" : "=r" (ret));
	return ret;
}

const struct bpf_func_proto bpf_read_64bit_xmm_register_proto = {
	.func		= bpf_read_64bit_xmm_register,
	.gpl_only	= false,
	.might_sleep	= false,
	.ret_type	= RET_INTEGER,
	.arg1_type	= ARG_ANYTHING,
};
```

GO GAUGES

```
var sub_fake_gauge = prometheus.NewGauge(prometheus.GaugeOpts{
        Name: "sub_fake_gauge",
        Help: "Subtracts 5 at every second",
})
<SNIP>
        go func() {
                for {
                        select {
                        case <-ticker.C:
                                another_fake_gauge.Add(5)
                                sub_fake_gauge.Sub(7)
                                dec_fake_gauge.Dec()
                        }
                }
        }()
```

USING IT

```
# ./prometheusnoop --include_description -b tests/prometheus/main
TIME     EVENT(Object)             PID
11:12:18 (0xc00021a3c0) 2431   : desc: "another_fake_gauge" value: 0.000000 inc: 5.000000
11:12:18 (0xc00021a440) 2431   : desc: "sub_fake_gauge" value: 0.000000 inc: 7.000000
11:12:18 (0xc00021a400) 2431   : desc: "dec_fake_gauge" value: 0.000000 inc: -7.000000
11:12:19 (0xc000218060) 2431   : desc: "fake_counter" value: 0
11:12:20 (0xc0002180c0) 2431   : desc: "another_fake_counter" value: 0
11:12:20 (0xc0002180c0) 2431   : desc: "another_fake_counter" value: 1
11:12:21 (0xc00021a380) 2431   : desc: "fake_gauge" value: 0.000000
11:12:22 (0xc00021a3c0) 2431   : desc: "another_fake_gauge" value: 5.000000 inc: 5.000000
11:12:22 (0xc00021a440) 2431   : desc: "sub_fake_gauge" value: -7.000000 inc: 7.000000
11:12:22 (0xc00021a400) 2431   : desc: "dec_fake_gauge" value: -1.000000 inc: -7.000000
11:12:23 (0xc000218060) 2431   : desc: "fake_counter" value: 1
11:12:24 (0xc0002180c0) 2431   : desc: "another_fake_counter" value: 2
11:12:24 (0xc0002180c0) 2431   : desc: "another_fake_counter" value: 3
11:12:25 (0xc00021a380) 2431   : desc: "fake_gauge" value: 1.000000
11:12:26 (0xc00021a3c0) 2431   : desc: "another_fake_gauge" value: 10.000000 inc: 5.000000
11:12:26 (0xc00021a440) 2431   : desc: "sub_fake_gauge" value: -14.000000 inc: 7.000000
11:12:26 (0xc00021a400) 2431   : desc: "dec_fake_gauge" value: -2.000000 inc: -7.000000
11:12:27 (0xc000218060) 2431   : desc: "fake_counter" value: 2
11:12:28 (0xc0002180c0) 2431   : desc: "another_fake_counter" value: 4
11:12:28 (0xc0002180c0) 2431   : desc: "another_fake_counter" value: 5
11:12:29 (0xc00021a380) 2431   : desc: "fake_gauge" value: 2.000000
11:12:30 (0xc00021a3c0) 2431   : desc: "another_fake_gauge" value: 15.000000 inc: 5.000000
11:12:30 (0xc00021a440) 2431   : desc: "sub_fake_gauge" value: -21.000000 inc: 7.000000
11:12:30 (0xc00021a400) 2431   : desc: "dec_fake_gauge" value: -3.000000 inc: -7.000000
^C#
```

HELPER

- Not just for x86_64
- Not just 64-bit registers
- Check if DWARF has generic registers for xmmN

LIMITATIONS

- Can only obtain the internal state
- So try to read it at function exit
- After increment?

URETPROBES CAN'T GO

- go changes the stack layout
- uretprobes don't like it
- (not) funny crashes
- delve seems to workaround this

PERF BENCH

- perf bench uprobes
- Measure baseline
- Then with a simple BPF attached
- Growing complexity

BENCHMARKS

```
# perf bench uprobe

        # List of available benchmarks for collection 'uprobe':

      baseline: Baseline libc usleep(1000) call
         empty: Attach empty BPF prog to uprobe on usleep, system wide
  trace_printk: Attach trace_printk BPF prog to uprobe on usleep syswide

#
```

RUNNING

```
# grep -m1 'model name' /proc/cpuinfo
model name      : Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz
```

RUNNING

```
# grep -m1 'model name' /proc/cpuinfo
model name      : Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz
```

```
# perf bench uprobe all
# Running uprobe/baseline benchmark...
# Executed 1,000 usleep(1000) calls
     Total time: 1,145,049 usecs

 1,145.049 usecs/op
```

RUNNING

```
# grep -m1 'model name' /proc/cpuinfo
model name      : Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz
```

```
# perf bench uprobe all
# Running uprobe/baseline benchmark...
# Executed 1,000 usleep(1000) calls
     Total time: 1,145,049 usecs

 1,145.049 usecs/op
```

```
# Running uprobe/empty benchmark...
# Executed 1,000 usleep(1000) calls
     Total time: 1,168,813 usecs +23,764 to baseline

 1,168.813 usecs/op 23.764 usecs/op to baseline
```

RUNNING

```
# grep -m1 'model name' /proc/cpuinfo
model name      : Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz


# perf bench uprobe all
# Running uprobe/baseline benchmark...
# Executed 1,000 usleep(1000) calls
     Total time: 1,145,049 usecs

 1,145.049 usecs/op


# Running uprobe/empty benchmark...
# Executed 1,000 usleep(1000) calls
     Total time: 1,168,813 usecs +23,764 to baseline

 1,168.813 usecs/op 23.764 usecs/op to baseline


# Running uprobe/trace_printk benchmark...
# Executed 1,000 usleep(1000) calls
     Total time: 1,173,868 usecs +28,819 to baseline +5,055 to previous

 1,173.868 usecs/op 28.819 usecs/op to baseline 5.055 usecs/op to previous

#
```

PERF BENCH TODO

- Add more benchmarks
- Start a workload
- Specifying which functions to put uprobes

MORE

- Play with the other non-C userspace
- Rust for both user and kernel space

RUST

- Miguel Ojeda asks for slides
- For Kangrejos Conference
- September 16-17, in Gijón, Asturias

PERF

```
$ perf probe -L rust_begin_unwind

      0  fn panic(info: &core::panic::PanicInfo<'_>) -> ! {
      1      pr_emerg!("{}\n", info);
             // SAFETY: FFI call.
      3      unsafe { bindings::BUG() };
         }
#
```

PERF

```
$ perf probe -L rust_begin_unwind

      0  fn panic(info: &core::panic::PanicInfo<'_>) -> ! {
      1      pr_emerg!("{}\n", info);
             // SAFETY: FFI call.
      3      unsafe { bindings::BUG() };
         }
#


# perf probe rust_begin_unwind
Failed to write event: Invalid argument
  Error: Failed to add events.
#
```

PERF

```
$ perf probe -L rust_begin_unwind

      0  fn panic(info: &core::panic::PanicInfo<'_>) -> ! {
      1      pr_emerg!("{}\n", info);
             // SAFETY: FFI call.
      3      unsafe { bindings::BUG() };
         }
#


# perf probe rust_begin_unwind
Failed to write event: Invalid argument
  Error: Failed to add events.
#


# dmesg | tail -1
[ 9771.947668] trace_kprobe: Could not probe notrace function _text
#
```

PAHOLE

```
# pahole rust/alloc.o -C \&str
die__process_class: tag not supported 0x33 (variant_part)!
die__process_function: tag not supported 0x2f (template_type_parameter)!
struct &str {
        u8 *         data_ptr __attribute__((__aligned__(8))); /* 0  8 */
        usize        length __attribute__((__aligned__(8)));   /* 8  8 */

        /* size: 16, cachelines: 1, members: 2 */
        /* forced alignments: 2 */
        /* last cacheline: 16 bytes */
} __attribute__((__aligned__(8)));
#
```

RUST DWARF

```
<1>: Abbrev Number: 8 (DW_TAG_structure_type)
      DW_AT_name        : (indirect string, offset: 0x840): &str
      DW_AT_byte_size   : 16
      DW_AT_alignment   : 8
<2>: Abbrev Number: 4 (DW_TAG_member)
      DW_AT_name        : (indirect string, offset: 0x830): data_ptr
      DW_AT_type        : <0xbe6>
      DW_AT_alignment   : 8
      DW_AT_data_member_location: 0
<2>: Abbrev Number: 4 (DW_TAG_member)
      DW_AT_name        : (indirect string, offset: 0x839): length
      DW_AT_type        : <0x89>
      DW_AT_alignment   : 8
      DW_AT_data_member_location: 8
```

THE END

- https://fedorapeople.org/~acme/prez/tracing-summit-2023
- https://perf.wiki.kernel.org/index.php/Useful_Links
- acme@kernel.org
- https://twitter.com/acmel