LOOKING AT YOURSELF

Linux introspection tales

Arnaldo Carvalho de Melo
acme@redhat.com

WHAT IS THIS ABOUT?

- Type information
- Introspection
- Adaptation

SHRINKING SOCKETS

- Linux 2.4
- struct sock
- Big union
- All protocols

SHRUNK SOCKET

- Reduce CPU cache utilization
- Remove alignment holes
- Reorder struct fields
- Combine some bitfields
- Demote others
- To better pack

SOCKET HIERARCHY

- struct sock
- struct tcp_sock
- struct tcp6_sock
- struct udp_sock
- And all the others

NEVERENDING STORY

```
Date: Mon Nov 15 11:02:37 2021 -0800
From: Eric Dumazet

Move sk_bind_phc next to sk_peer_lock to fill a hole.

@@ -489,5 +489,6 @@  struct sock {
        u16                     sk_busy_poll_budget;
 #endif
        spinlock_t              sk_peer_lock;
+       int                     sk_bind_phc;
        struct pid              *sk_peer_pid;
        const struct cred       *sk_peer_cred;
@@ -498,5 +499,4 @@  struct sock {
        seqlock_t               sk_stamp_seq;
 #endif
        u16                     sk_tsflags;
-       int                     sk_bind_phc;
        u8                      sk_shutdown;
```

SOME MORE

```
$ git log -2 --oneline 1ace2b4d2b4e1db8f
1ace2b4d2b4e1db8 net: shrink struct sock by 8 bytes
1b31debca8328448 ipv6: shrink struct ipcm6_cookie
$
```

TEDIOUS

- Manual
- Can we automate this?
- gdb knows about types, how?

DWARF

- Debugging With Arbitrary Record Formats
- Executable and Linkable Format's friend
- More recently we got ORC too

PAHOLE

- Read DWARF
- Rebuild source from type info
- Augmented with alignment info
- Showing the holes
- Cacheline boundaries
- Lots more

EXAMPLE

```
$ pahole -C list_head ~/git/build/v5.18-rc6+/vmlinux
struct list_head {
        struct list_head *         next;                 /*     0
        struct list_head *         prev;                 /*     8

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

CERN ATLAS MIGRATION

- 32-bit to 64-bit
- C++
- long/pointer: 32 to 64-bit

AN ALIGNMENT HOLE

```
# pahole -C _IO_FILE ~/bin/perf | head
struct _IO_FILE {
        int                        _flags;               /*     0

        /* XXX 4 bytes hole, try to pack */

        char *                     _IO_read_ptr;         /*     8
        char *                     _IO_read_end;         /*    16
        char *                     _IO_read_base;        /*    24
        char *                     _IO_write_base;       /*    32
        char *                     _IO_write_ptr;        /*    40
#
```

PAHOLE --REORGANIZE

- To elliminate holes
- Demotes bitfields
- show-reorg-steps
- __alignment__ attribute
- false sharing

PAHOLE --REORGANIZE 2

- DWARF now has attribute align
- Use that in the reorg algo

--REORGANIZE EXAMPLE

```
$ pahole --reorganize task_struct | tail
    /* --- cacheline 142 boundary (9088 bytes) was 56 bytes ago ---
    struct thread_struct thread __attribute__((__aligned__(64))); /

    /* size: 13560, cachelines: 212, members: 252 */
    /* sum members: 13487, holes: 2, sum holes: 57 */
    /* sum bitfield members: 79 bits, bit holes: 2, sum bit holes: 4
    /* paddings: 6, sum paddings: 49 */
    /* forced alignments: 2, forced holes: 1, sum forced holes: 56
    /* last cacheline: 56 bytes */
};  /* saved 136 bytes and 2 cachelines! */
$
```

CTF

- Dtrace
- Solaris
- In the kernel image
- Introspection
- SparcLinux

MULTI-FORMAT

- Type info agnostic
- DWARF and CTF
- CTF reader
- CTF encoder

CONVERSION

- From DWARF to CTF
- For testing

DWARF PROBLEMS

- kernel community problems with unwinding
- object files: Compile Units
- All types represented per CU
- debuginfo files are big
- hundreds of megabytes

BPF NEEDS TYPE INFO

- To pretty print maps
- Later: CO-RE

BTF

- BPF Type Format
- Reused the CTF infra in pahole
- First BTF producer: clang bpf target
- First BTF consumer: Linux kernel

BPF DEDUP

- Looks at all objects
- Removes type duplicates

BTF READER

- For testing
- /sys/kernel/btf/vmlinux
- Wow, that is fast!

PAHOLE + BTF

- Its always there
- So inform just the type

PAHOLE + BTF

```
$ pahole spinlock_t
typedef struct spinlock spinlock_t;
$
```

PAHOLE + BTF

```
$ pahole spinlock
struct spinlock {
        union {
                struct raw_spinlock rlock;              /*      0
        };                                              /*      0

        /* size: 4, cachelines: 1, members: 1 */
        /* last cacheline: 4 bytes */
};
$
```

EXPAND IT!

```
$ pahole -E spinlock
struct spinlock {
    union {
        struct raw_spinlock {
            /* typedef arch_spinlock_t */ struct qspinlock {
                union {
                    /* typedef atomic_t */ struct {
                        int            counter;                                  /* 0  4 */
                    } val;                                                       /* 0  4 */
                    struct {
                        /* typedef u8 -> __u8 */ unsigned char  locked;          /* 0  1 */
                        /* ypedef u8 -> __u8 */ unsigned char  pending;          /* 1  1 */
                    };                                                           /* 0  2 */
                    struct {
                        /* typedef u16 -> __u16 */ short unsigned int locked_pending; /* 0  2 */
                        /* typedef u16 -> __u16 */ short unsigned int tail;      /* 2  2 */
                    };                                                           /* 0  4 */
                };                                                               /* 0  4 */
            } raw_lock;                                                          /* 0  4 */
        } rlock;                                                                 /* 0  4 */
    };                                                                           /* 0  4 */
    /* size: 4, cachelines: 1, members: 1 */
    /* last cacheline: 4 bytes */
};
$
```

WHERE IS IT?

- libbpf loads it
- kernel verifies
- Gets associated to the prog/map fd
- Tools can retrieve it

BPFTOOL BPF

- Generates vmlinux.h
- With all kernel types

```
bpftool btf dump file /sys/kernel/btf/vmlinux format c
```

PAHOLE --COMPILE

- Reconstruct compilable source code
- Like bpftool

WHERE ELSE?

- /sys/kernel/btf/
- vmlinux and modules
- modules refer to types in vmlinux

PROGRAM LINES

- .BTF_ext ELF section
- perf annotate
- BPF source code

ADAPTABILITY: USE CASE

```
Author: Namhyung Kim
Date:   Wed May 18 15:47:23 2022 -0700

  perf record: Handle argument change in sched_switch

  Recently sched_switch tracepoint added a new argument for
  prev_state, but it's hard to handle the change in a BPF
  program.  Instead, we can check the function prototype in
  BTF before loading the program.
```

ADAPTABILITY: TOOL LOADER

```
static void check_sched_switch_args(void)
{
        struct btf *btf = bpf_object__btf(skel->obj);
        struct btf_type *t1, *t2, *t3;
        u32 type_id = btf__find_by_name_kind(btf, "bpf_trace_sched_sw
                                             BTF_KIND_TYPEDEF);
        t1 = btf__type_by_id(btf, type_id);
        t2 = btf__type_by_id(btf, t1->type);

        if (t3 && btf_is_func_proto(t3) && btf_vlen(t3) == 4) {
                // new format: pass prev_state as 4th arg
                skel->rodata->has_prev_state = true;
        }
}
```

ADAPTABILITY: BPF SKEL CHANGE

```
+++ b/tools/perf/util/bpf_skel/off_cpu.bpf.c

+const volatile bool has_prev_state = false;

+SEC("tp_btf/sched_switch")
+int on_switch(u64 *ctx) {
+       struct task_struct *prev, *next;
+       int prev_state;
+
+       if (!enabled) return 0;

+       prev = (struct task_struct *)ctx[1];
+       next = (struct task_struct *)ctx[2];
+
+       if (has_prev_state)
+               prev_state = (int)ctx[3];
+       else
+               prev_state = get_task_state(prev);
+       return off_cpu_stat(ctx, prev, next, prev_state);
+}
```

RUNNING IT

```
# perf record --off-cpu
^C[ perf record: Woken up 1924 times to write data ]
[ perf record: Captured and wrote 483.936 MB perf.data (8857075 sam

# ls -la perf.data
-rw-------. 1 root root 507845510 May 31 00:45 perf.data
#
```

OBSERVING IT

```
# bpftool prog | grep on_switch -A4
634: tracing  name on_switch  tag 3d6d5a513a933c28  gpl
  loaded_at 2022-05-30T22:37:17+0200  uid 0
  xlated 1392B jited 913B memlock 4096B map_ids 497,498,493,494,495
  btf_id 602
  pids perf(393176)
#
```

SHOW ME THE CODE

```
# bpftool prog dump jited id 634
int on_switch(u64 * ctx):
bpf_prog_3d6d5a513a933c28_on_switch:
; int on_switch(u64 *ctx)
   0:   nopl   0x0(%rax,%rax,1)
   5:   xchg   %ax,%ax
   7:   push   %rbp
   8:   mov    %rsp,%rbp
   b:   sub    $0x38,%rsp
  12:   push   %rbx
  13:   push   %r13
  15:   push   %r14
  17:   push   %r15
  19:   mov    %rdi,%r15
; if (!enabled)
  1c:   movabs $0xffffb53a400d2000,%rdi
  26:   mov    0x0(%rdi),%edi
; if (!enabled)
  29:   test   %rdi,%rdi
  2c:   je     0x0000000000000386
```

SHOW ME THE CODE: 2

```
; next = (struct task_struct *)ctx[2];
  32:   mov    0x10(%r15),%r14
; prev = (struct task_struct *)ctx[1];
  36:   mov    0x8(%r15),%rbx
; if (has_prev_state)
  3a:   movabs $0xffffb53a400f6000,%rdi
  44:   movzbq 0x0(%rdi),%rdi
; prev_state = (int)ctx[3];
  49:   mov    $0x1,%edi
; if (bpf_core_field_exists(t->__state))
  4e:   mov    $0x18,%edi
  53:   mov    %rbx,%rdx
  56:   add    %rdi,%rdx
  59:   mov    %rbp,%rdi
;
  5c:   add    $0xffffffffffffffd8,%rdi
; return BPF_CORE_READ(t, __state);
  60:   mov    $0x4,%esi
  65:   callq  0xffffffffd5f13f50
; return BPF_CORE_READ(t, __state);
  6a:   mov    -0x28(%rbp),%r13d
```

WHERE ARE THE MAPS

```
# bpftool map | grep off_cpu -A3
490: array  name off_cpu_.rodata  flags 0x480
        key 4B  value 3B  max_entries 1  memlock 4096B
        btf_id 628  frozen
        pids perf(393176)
#
```

ITS CONTENTS

```
# bpftool map dump id 490
[{
        "value": {
            ".rodata": [{
                    "has_prev_state": false
                },{
                    "needs_cgroup": false
                },{
                    "uses_cgroup_v1": false
                }
            ]
        }
    }
#
```

THE FEATURE

```
# perf report --stdio --call-graph=no
# Childr  Self Command          Shared Object       Symbol
# ...... ...... ............... ..................... ................
  81.66%  0.00% sched-messaging libc-2.33.so         [.] __libc_start_
  81.66%  0.00% sched-messaging perf                 [.] cmd_bench
  81.66%  0.00% sched-messaging perf                 [.] main
  81.66%  0.00% sched-messaging perf                 [.] run_builtin
  81.43%  0.00% sched-messaging perf                 [.] bench_sched_m
  40.86% 40.86% sched-messaging libpthread-2.33.so   [.] __read
  37.66% 37.66% sched-messaging libpthread-2.33.so   [.] __write
   2.91%  2.91% sched-messaging libc-2.33.so         [.] __poll
...
```

As you can see it spent most of off-cpu time in read and write i
bench_sched_messaging(). The --call-graph=no was added just to
the output concise here.

perf annotate

```
Samples: 6M of events 'anon group { cycles:P, L1-icache-load-misses }', 4000 Hz, Event count (approx.): 42093638433
bpf_prog_0bc3fc9d11754ba1_sys_enter  bpf_prog_0bc3fc9d11754ba1_sys_enter [Percent: local period]
Percent
 54.68   2.34       int sys_enter(struct syscall_enter_args *args)
  0.71  68.27         push   %rbp
  0.23   0.20         mov    %rsp,%rbp
  6.46   0.11         sub    $0x200,%rsp
  0.55  13.91         push   %rbx
  0.46   1.14         push   %r13
  0.33   1.12         push   %r14
  0.55   0.55         push   %r15
  0.03   1.38         pushq  $0x0
                      mov    %rdi,%rbx
                         return bpf_get_current_pid_tgid();
  0.54   0.15       → callq  *ffffffffe017a907
  1.41   0.04         mov    %eax,-0x8(%rbp)
  0.02   3.20         mov    %rbp,%rsi

                      add    $0xfffffffffffffff8,%rsi
                         return bpf_map_lookup_elem(pids, &pid) != NULL;
  0.11   0.18         movabs $0xffff8b6b8418c800,%rdi
  1.28   0.02       → callq  *ffffffffe017c2a7
  0.02   0.21         cmp    $0x0,%rax
  0.25   0.00       ↓ je     3f
  0.10   0.05         add    $0x38,%rax
  0.54   0.87  3f:   mov    %rax,%rdi
  0.02   0.43         xor    %eax,%eax
                         if (pid_filter__has(&pids_filtered, getpid()))
  0.05   0.00         cmp    $0x0,%rdi
                    ↓ jne    db
  0.55   0.12         xor    %edi,%edi
Press 'h' for help on key bindings
```

nospectre_v1 + nospectre_v2

```
Samples: 954K of events 'anon group { cycles:P, L1-icache-load-misses }', 4000 Hz, Event count (approx.): 77077306917
bpf_prog_0bc3fc9d11754ba1_sys_enter  bpf_prog_0bc3fc9d11754ba1_sys_enter [Percent: local period]
Percent                 int sys_enter(struct syscall_enter_args *args)
 12.07   14.45            push   %rbp
  0.06   14.93            mov    %rsp,%rbp
  6.19    0.00            sub    $0x200,%rsp
  1.72    3.90            push   %rbx
  0.97    3.70            push   %r13
  0.00    0.21            push   %r14
  2.08    0.76            push   %r15
  0.00    4.80            pushq  $0x0
  0.00    1.03            mov    %rdi,%rbx
                             return bpf_get_current_pid_tgid();
  8.76    0.00          → callq  *ffffffffd7d26567
  8.13    3.65            mov    %eax,-0x8(%rbp)
  0.00   11.17            mov    %rbp,%rsi

                          add    $0xffffffffffffffff8,%rsi
                             return bpf_map_lookup_elem(pids, &pid) != NULL;
  2.08    0.00            movabs $0xffff9a6f1c0d9400,%rdi
  1.18    0.78          → callq  *ffffffffd7d27f07
  0.16    0.00            cmp    $0x0,%rax
  0.27    0.00          ↓ je     3f
  0.00    0.65            add    $0x38,%rax
  2.76    1.18  3f:       mov    %rax,%rdi
  0.00    9.61            xor    %eax,%eax
                             if (pid_filter__has(&pids_filtered, getpid()))
  0.21    0.31            cmp    $0x0,%rdi
  0.66    0.00          ↓ jne    db
  0.62    0.84            xor    %edi,%edi
                             int key = 0;
  0.57    1.02            mov    %edi,-0x4(%rbp)
  0.00    0.69            mov    %rbp,%rsi

                          add    $0xfffffffffffffffc,%rsi
                             return bpf_map_lookup_elem(&augmented_args_tmp, &key);
Press 'h' for help on key bindings
```

PAHOLE IN THE KERNEL BUILD

```
LD [M]  drivers/media/usb/gspca/gspca_zc3xx.o
AR      drivers/media/built-in.a
AR      drivers/built-in.a
GEN     .version
CHK     include/generated/compile.h
LD      vmlinux.o
MODPOST vmlinux.symvers
MODINFO modules.builtin.modinfo
GEN     modules.builtin
CC      .vmlinux.export.o
LD      .tmp_vmlinux.btf
BTF     .btf.vmlinux.bin.o
LD      .tmp_vmlinux.kallsyms1
KSYMS   .tmp_vmlinux.kallsyms1.S
AS      .tmp_vmlinux.kallsyms1.S
LD      .tmp_vmlinux.kallsyms2
KSYMS   .tmp_vmlinux.kallsyms2.S
AS      .tmp_vmlinux.kallsyms2.S
LD      vmlinux
```

MODULES TOO

```
LD [M]  arch/x86/kvm/kvm-amd.ko
BTF [M] arch/x86/kernel/cpu/mce/mce-inject.ko
BTF [M] arch/x86/events/rapl.ko
LD [M]  arch/x86/kvm/kvm-intel.ko
LD [M]  arch/x86/kvm/kvm.ko
BTF [M] arch/x86/kvm/kvm-amd.ko
LD [M]  crypto/adiantum.ko
BTF [M] crypto/adiantum.ko
BTF [M] arch/x86/kvm/kvm-intel.ko
LD [M]  crypto/aegis128.ko
BTF [M] crypto/aegis128.ko
LD [M]  crypto/aes_ti.ko
BTF [M] arch/x86/kvm/kvm.ko
```

NEW BPF/BTF FEATURES

- pahole as an enabler
- Eventually compilers will produce BTF
- But you have to dedup when generating vmlinux
- Generate BTF info for VARs after linking
- Covenient when developing new features

BTF TAGS

- clang generates new DWARF tags
- pahole converts to BTF_KIND_TAG
- kernel BPF verifier uses
- __rcu, percpu, etc

KERNEL BUILD PAHOLE CAPABILITY QUERY

```
$ cat scripts/pahole-flags.sh
#!/bin/sh
if [ "${pahole_ver}" -ge "118" ] && [ "${pahole_ver}" -le "121" ]; t
        # pahole 1.18 through 1.21 can't handle zero-sized per-CPU v
        extra_paholeopt="${extra_paholeopt} --skip_encoding_btf_var
fi
if [ "${pahole_ver}" -ge "121" ]; then
        extra_paholeopt="${extra_paholeopt} --btf_gen_floats"
fi
if [ "${pahole_ver}" -ge "122" ]; then
        extra_paholeopt="${extra_paholeopt} -j"
fi

echo ${extra_paholeopt}
$ scripts/pahole-flags.sh
--btf_gen_floats -j
$
pahole --version
v1.23
$
```

RUST

- reordering of struct fields
- pahole --reorganize done by rust
- rejected by kernel BTF verifier
- kernel build: pahole --lang_exclude rust
- pahole should put fields in order instead
- when generating BTF for rust kernel objects
- DWARF compile_unit tag has DW_AT_producer

DWARF LANGS

```
static const char *languages[] = {
  [DW_LANG_Ada83]          = "ada83",
SNIP
  [DW_LANG_C11]            = "c11",
  [DW_LANG_C89]            = "c89",
  [DW_LANG_C99]            = "c99",
  [DW_LANG_C]              = "c",
  [DW_LANG_Cobol74]        = "cobol74",
SNIP
  [DW_LANG_C_plus_plus_14] = "c++14",
  [DW_LANG_C_plus_plus]    = "c++",
  [DW_LANG_D]              = "d",
  [DW_LANG_Dylan]          = "dylan",
  [DW_LANG_Fortran03]      = "fortran03",
SNIP
  [DW_LANG_PLI]            = "pli",
  [DW_LANG_Python]         = "python",
  [DW_LANG_RenderScript]   = "renderscript",
  [DW_LANG_Rust]           = "rust",
```

```
};
```

PRETTY PRINTING IN THE KERNEL

- bpf_seq_printf_btf
- bpf_snprintf_btf BPF helper

BPF_SNPRINTF_BTF HELPER

```
static struct btf_ptr b = { };

b.ptr = skb;
b.type_id = __builtin_btf_type_id(struct sk_buff, 1);
bpf_snprintf_btf(str, sizeof(str), &b, sizeof(b), 0, 0)
```

BPF_SNPRINTF_BTF HELPER

Default output looks like this:

```
(struct sk_buff){
 .transport_header = (__u16)65535,
 .mac_header = (__u16)65535,
 .end = (sk_buff_data_t)192,
 .head = (unsigned char *)0x000000007524fd8b,
 .data = (unsigned char *)0x000000007524fd8b,
 .truesize = (unsigned int)768,
 .users = (refcount_t){
  .refs = (atomic_t){
   .counter = (int)1,
  },
 },
}
```

FLAGS

```
Flags modifying display are as follows:

- BTF_F_COMPACT:        no formatting around type information
- BTF_F_NONAME:         no struct/union member names/types
- BTF_F_PTR_RAW:        show raw (unobfuscated) pointer values;
                        equivalent to %px.
- BTF_F_ZERO:           show zero-valued struct/union members;
                        they are not displayed by default
```

PAHOLE PRETTY PRINTING

```
$ pahole --prettify=- --header elf64_hdr < /bin/bash
{
        .e_ident = { 127, 69, 76, 70, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0,
        .e_type = 3,
        .e_machine = 62,
        .e_version = 1,
        .e_entry = 204224,
        .e_phoff = 64,
        .e_shoff = 1388016,
        .e_flags = 0,
        .e_ehsize = 64,
        .e_phentsize = 56,
        .e_phnum = 13,
        .e_shentsize = 64,
        .e_shnum = 32,
        .e_shstrndx = 31,
},
$
```

ITS IN THE KERNEL BTF

```
$ pahole elf64_hdr
struct elf64_hdr {
        unsigned char             e_ident[16];          /*     0
        Elf64_Half                e_type;               /*    16
        Elf64_Half                e_machine;            /*    18
        Elf64_Word                e_version;            /*    20
        Elf64_Addr                e_entry;              /*    24
        Elf64_Off                 e_phoff;              /*    32
        Elf64_Off                 e_shoff;              /*    40
        Elf64_Word                e_flags;              /*    48
        Elf64_Half                e_ehsize;             /*    52
        Elf64_Half                e_phentsize;          /*    54
        Elf64_Half                e_phnum;              /*    56
        Elf64_Half                e_shentsize;          /*    58
        Elf64_Half                e_shnum;              /*    60
        Elf64_Half                e_shstrndx;           /*    62

        /* size: 64, cachelines: 1, members: 14 */
};
$
```

BTFGEN

- For older kernels
- Generates kernel eBPF needed by a prog
- Modified libbpf looks at it as the kernel BPF
- Does its CO-RE work

TETRAGON

- Cilium
- Security
- BTF required
- N-level filtering

THE END

- http://vger.kernel.org/~acme/prez/kernel-recipes-202
- BTF dedup: https://facebookmicrosites.github.io/bpf/blog/2018/1 enhancement.html>
- BTF CO-RE: https://facebookmicrosites.github.io/bpf/blog/2020/0 portability-and-co-re.html
- acme@kernel.org
- https://twitter.com/acmel