

20 YEARS OF PAHOLE: ALIVE AND KICKING!

Arnaldo Carvalho de Melo

Alan Maguire

Linux Plumbers Conference 2026

Refereed Track



WHAT IS THIS ABOUT?

- 20 years: from struct packing to BTF
- DWARF is a moving target
- Testing strategies that survive distros
- What's new: C++, Rust, dwz, coverage
- perf integration: becoming real
- BTF: recent developments

OCTOBER 24, 2006

- First cset, twenty years ago
- Goal: reorganize kernel networking data structures
- More than first hoped
- Many projects view structs with precision

IN THE KERNEL GIT HISTORY

- Nov 27, 2006: first recorded use
- One month after the first cset
- 603 commits quote pahole or its telltales
- 482 name pahole, 121 only its output
- Since 2019: `kbuild BTF` with pahole

FIRST RECORDED USE: NOV 27, 2006

```
$ pahole net/ipv6/tcp_ipv6.o inet_connection_sock_af_ops
struct inet_connection_sock_af_ops {
    int (*queue_xmit)();          /*      0      4 */
    ...
    __u16 net_header_len;         /*     24     2 */

    /* XXX 2 bytes hole, try to pack */

    int (*setsockopt)();          /*     28     4 */
    ...
}; /* size: 52, sum members: 50, holes: 1, sum holes: 2 */

$ codiff -sV /tmp/tcp_ipv6.o.before net/ipv6/tcp_ipv6.o
struct inet_connection_sock_af_ops |    -4
```

- sockaddr_len int → u16: 4 bytes saved
- **codiff** verifies, byte by byte

SWISS ARMY KNIFE

- DWARF to CTF conversion
- Most used now: DWARF to BTF
- Part of the kernel build: BPF CO-RE
- Still growing, 20 years on

WHO BUILT IT

- ~2,480 commits, 83 contributors, 20 years
- New faces every year: 5 in 2006, 10 in 2026
- BTF encoder: Meta (Martin, Andrii, Yonghong)
- Co-maintainer Alan Maguire (Oracle): BTF
- ARM, IBM, Google, Debian, kernel folks: each an itch

MULTI-FORMAT: CTF, THEN BTF

- Mar 2008: CTF, on DaveM's request (his libctf)
- Made pahole multi-format
- BTF: discussed with Alexei and Daniel Borkmann, Santa Fe, 2015
- Martin KaFai Lau (Meta) contributed it, Mar 2018
- First use: pretty-print BPF maps
- Now: kernel build, CO-RE

TYPES DECODE FILES: --PRETTIFY

```
$ pahole --header elf64_hdr --prettify=/bin/bash
{
    .e_ident = { 127, 69, 76, 70, 2, 1, 1, 0, ... },
    .e_type = 3,
    .e_machine = 62,
    .e_entry = 204016,
    ...
},
```

- Types describe the file: ELF, perf.data, ...

PERF.DATA, BY ITS OWN TYPES

```
1 $ pahole -F dwarf ~/bin/perf --header=perf_file_header \  
2   --seek_bytes '$header.data.offset' --size_bytes='$header.data.size' \  
3   -C 'perf_event_header(sizeof,type,type_enum=perf_event_type,filter=type==PERF_RECORD_MMAP2)' \  
4   --prettify perf.data --count 1  
5 {  
6     .header = {  
7         .type = PERF_RECORD_MMAP2,  
8         .misc = 2,  
9         .size = 112,  
10    },  
11    .pid = 1533617,  
12    .tid = 1533617,  
13    .start = 94667542700032,  
14    .len = 90112,  
15    ...  
16    .filename = "/usr/bin/ls",  
17 },
```

- perf.data types come from the perf binary itself
- Record type picks the struct: MMAP2 here

PERF.DATA, BY ITS OWN TYPES

```
1 $ pahole -F dwarf ~/bin/perf --header=perf_file_header \  
2   --seek_bytes '$header.data.offset' --size_bytes='$header.data.size' \  
3   -C 'perf_event_header(sizeof,type,type_enum=perf_event_type,filter=type==PERF_RECORD_MMAP2)' \  
4   --prettify perf.data --count 1  
5 {  
6     .header = {  
7         .type = PERF_RECORD_MMAP2,  
8         .misc = 2,  
9         .size = 112,  
10    },  
11    .pid = 1533617,  
12    .tid = 1533617,  
13    .start = 94667542700032,  
14    .len = 90112,  
15    ...  
16    .filename = "/usr/bin/ls",  
17 },
```

- perf.data types come from the perf binary itself
- Record type picks the struct: MMAP2 here

PERF.DATA, BY ITS OWN TYPES

```
1 $ pahole -F dwarf ~/bin/perf --header=perf_file_header \  
2   --seek_bytes '$header.data.offset' --size_bytes='$header.data.size' \  
3   -C 'perf_event_header(sizeof,type,type_enum=perf_event_type,filter=type==PERF_RECORD_MMAP2)' \  
4   --prettify perf.data --count 1  
5 {  
6     .header = {  
7         .type = PERF_RECORD_MMAP2,  
8         .misc = 2,  
9         .size = 112,  
10    },  
11    .pid = 1533617,  
12    .tid = 1533617,  
13    .start = 94667542700032,  
14    .len = 90112,  
15    ...  
16    .filename = "/usr/bin/ls",  
17 },
```

- perf.data types come from the perf binary itself
- Record type picks the struct: MMAP2 here

PERF.DATA, BY ITS OWN TYPES

```
1 $ pahole -F dwarf ~/bin/perf --header=perf_file_header \  
2   --seek_bytes '$header.data.offset' --size_bytes='$header.data.size' \  
3   -C 'perf_event_header(sizeof,type,type_enum=perf_event_type,filter=type==PERF_RECORD_MMAP2)' \  
4   --prettify perf.data --count 1  
5 {  
6     .header = {  
7         .type = PERF_RECORD_MMAP2,  
8         .misc = 2,  
9         .size = 112,  
10    },  
11    .pid = 1533617,  
12    .tid = 1533617,  
13    .start = 94667542700032,  
14    .len = 90112,  
15    ...  
16    .filename = "/usr/bin/ls",  
17 },
```

- perf.data types come from the perf binary itself
- Record type picks the struct: MMAP2 here

COMPILABLE OUTPUT: FULLCIRCLE

```
$ pfunct --compile file.o > file.c    # C from DWARF
$ CFLAGS=$(readelf -wi file.o | grep DW_AT_producer ...)
$ gcc $CFLAGS -c -g file.c -o file2.o
$ codiff -q -s file.o file2.o        # silent: same types
```

- Binary → C → binary: types survive the round trip
- Script fullcircle in the pahole tree, used in Cl

DWARF IS A MOVING TARGET

- Compilers: gcc, clang, rustc...
- Versions, vendors, distro packaging
- Standards evolve: [DWARF 2..5](#), 6 coming
- Same source, different DWARF

DWARF DECIDES WHAT YOU SEE

```
typedef _Atomic int atomic_int;  
struct test { atomic_int x; };
```

```
gcc -gdwarf-5 -c test.c
```

- clang 21.1.8: typedef → base_type, no atomic node
- gcc/clang 22+: typedef → DW_TAG_atomic_type
- Both correct: pahole reads DWARF, not source
- DWARF version varies by distro: force -gdwarf-5
- Proof: tests/tests -v --dump-artifacts 51

REGRESSION TESTS: PASS OR SKIP

- Pass or gracefully skip: RHEL8 to Fedora latest
- Detect capabilities, don't assume them
- Every skip explains: what, where, why not a bug
- ~77 tests in [tests/](#), 95 commits since Jan 2025
- With AI coding agents

COVERAGE, REGULARLY

```
$ scripts/coverage_table.py # llvm-cov, best first
```

	# hits	# misses	# branches	coverage
dwarf_loader.c	3376	2559	817	75.8%
pahole.c	2178	1465	713	67.3%
TOTAL	17426	11599	5827	66.6%

(container run: perf tests skipped, 74 0k)

- `coverage/coverage-diff.sh`: what a change moves

LANGUAGES: C++ SINCE 2006

- Dec 2006: C++, weeks after the Oct 24 start
- Early users: CERN, moving 32 to 64-bit
- Then: namespaces, `class_type`, templates
- Recent: C++ using-decls, more templates
- Rust: variant parts ([eBPF track talk!](#))
- Go: Trace Summit Bilbao 2023

WHAT'S NEW: PLUMBING

- DWZ alternate debug files
- Partial units, split DWARF
- Modern distro DWARF just works
- Coverage analysis, growing regression suite

--REORGANIZE: NAÏVE PACKING

```
$ pahole -R --show_reorg_steps -C demo demo.o
/* Moving 'd' from after 'c' to after 'b' */
struct demo {                /* was 32 bytes, now 24 */
    int a;                   /* 0 4 */
    char b;                  /* 4 1 */
    char d;                  /* 5 1 */
    long c;                  /* 8 8 */
    int e;                   /* 20 4 */
};
```

- Packs by size, kills holes. Knows nothing about use.
- Splits fields that travel together
- Packs fields that contend
- Open task: feed co-access + false-sharing into it

DATA TYPES, BY CACHELINE

```
1 # perf mem record -- perf test -w false_sharing
2 # perf report -s type,typecln -H --group --stdio
3 #           Overhead  Data Type / Data Type Cacheline
4    88.14%  32.83%      struct net_conn
5     58.96%  16.09%      struct net_conn: cache-line 0
6     29.18%   0.00%      struct net_conn: cache-line 2
7     0.00%  16.74%      struct net_conn: cache-line 1
```

- **perf mem**: Intel PEBS, AMD IBS, arm64 SPE
- **--group**: loads left, stores right
- **cache-line 0**: loads and stores
- **cache-line 1**: stores; **cache-line 2**: loads

DATA TYPES, BY CACHELINE

```
1 # perf mem record -- perf test -w false_sharing
2 # perf report -s type,typecln -H --group --stdio
3 #           Overhead  Data Type / Data Type Cacheline
4      88.14%  32.83%      struct net_conn
5      58.96%  16.09%      struct net_conn: cache-line 0
6      29.18%   0.00%      struct net_conn: cache-line 2
7       0.00%  16.74%      struct net_conn: cache-line 1
```

- **perf mem**: Intel PEBS, AMD IBS, arm64 SPE
- **--group**: loads left, stores right
- **cache-line 0**: loads and stores
- **cache-line 1**: stores; **cache-line 2**: loads

DATA TYPES, BY CACHELINE

```
1 # perf mem record -- perf test -w false_sharing
2 # perf report -s type,typecln -H --group --stdio
3 #           Overhead  Data Type / Data Type Cacheline
4    88.14%  32.83%      struct net_conn
5    58.96%  16.09%      struct net_conn: cache-line 0
6    29.18%   0.00%      struct net_conn: cache-line 2
7     0.00%  16.74%      struct net_conn: cache-line 1
```

- **perf mem**: Intel PEBS, AMD IBS, arm64 SPE
- **--group**: loads left, stores right
- **cache-line 0**: loads and stores
- **cache-line 1**: stores; **cache-line 2**: loads

DATA TYPES, BY CACHELINE

```
1 # perf mem record -- perf test -w false_sharing
2 # perf report -s type,typecln -H --group --stdio
3 #           Overhead  Data Type / Data Type Cacheline
4    88.14%  32.83%      struct net_conn
5    58.96%  16.09%      struct net_conn: cache-line 0
6    29.18%   0.00%      struct net_conn: cache-line 2
7     0.00%  16.74%      struct net_conn: cache-line 1
```

- **perf mem**: Intel PEBS, AMD IBS, arm64 SPE
- **--group**: loads left, stores right
- **cache-line 0**: loads and stores
- **cache-line 1**: stores; **cache-line 2**: loads

DATA TYPES, BY CACHELINE

```
1 # perf mem record -- perf test -w false_sharing
2 # perf report -s type,typecln -H --group --stdio
3 #           Overhead  Data Type / Data Type Cacheline
4      88.14%  32.83%      struct net_conn
5      58.96%  16.09%      struct net_conn: cache-line 0
6      29.18%   0.00%      struct net_conn: cache-line 2
7       0.00%  16.74%      struct net_conn: cache-line 1
```

- **perf mem**: Intel PEBS, AMD IBS, arm64 SPE
- **--group**: loads left, stores right
- **cache-line 0**: loads and stores
- **cache-line 1**: stores; **cache-line 2**: loads

STATIC VIEW: PAHOLE

```
1 $ pahole -C net_conn ~/bin/perf
2 struct net_conn {
3     uint32_t          saddr;           /*      0      4 */
4     uint32_t          daddr;           /*      4      4 */
5     uint16_t          sport;           /*      8      2 */
6     uint16_t          dport;           /*     10      2 */
7     uint8_t           state;           /*     12      1 */
8     uint8_t           protocol;        /*     13      1 */
9     uint64_t          bytes_rx;         /*     16      8 */
10    uint64_t          packets_rx;       /*     24      8 */
11    uint32_t          rx_queue;         /*     32      4 */
12    uint32_t          last_ack;         /*     60      4 */
13    /* --- cacheline 1 boundary (64 bytes) --- */
14    ...
15    /* size: 192, cachelines: 3, members: 26 */
16 } __attribute__((__aligned__(64)));
```

- Where each member sits, cacheline by cacheline
- No idea which members are hot

STATIC VIEW: PAHOLE

```
1 $ pahole -C net_conn ~/bin/perf
2 struct net_conn {
3     uint32_t      saddr;           /*      0      4 */
4     uint32_t      daddr;           /*      4      4 */
5     uint16_t      sport;           /*      8      2 */
6     uint16_t      dport;           /*     10      2 */
7     uint8_t       state;           /*     12      1 */
8     uint8_t       protocol;        /*     13      1 */
9     uint64_t      bytes_rx;        /*     16      8 */
10    uint64_t      packets_rx;       /*     24      8 */
11    uint32_t      rx_queue;         /*     32      4 */
12    uint32_t      last_ack;         /*     60      4 */
13    /* --- cacheline 1 boundary (64 bytes) --- */
14    ...
15    /* size: 192, cachelines: 3, members: 26 */
16 } __attribute__((__aligned__(64)));
```

- Where each member sits, cacheline by cacheline
- No idea which members are hot

STATIC VIEW: PAHOLE

```
1 $ pahole -C net_conn ~/bin/perf
2 struct net_conn {
3     uint32_t      saddr;           /*      0      4 */
4     uint32_t      daddr;           /*      4      4 */
5     uint16_t      sport;           /*      8      2 */
6     uint16_t      dport;           /*     10      2 */
7     uint8_t       state;           /*     12      1 */
8     uint8_t       protocol;        /*     13      1 */
9     uint64_t      bytes_rx;         /*     16      8 */
10    uint64_t      packets_rx;       /*     24      8 */
11    uint32_t      rx_queue;         /*     32      4 */
12    uint32_t      last_ack;         /*     60      4 */
13    /* --- cacheline 1 boundary (64 bytes) --- */
14    ...
15    /* size: 192, cachelines: 3, members: 26 */
16 } __attribute__((__aligned__(64)));
```

- Where each member sits, cacheline by cacheline
- No idea which members are hot

STATIC VIEW: PAHOLE

```
1 $ pahole -C net_conn ~/bin/perf
2 struct net_conn {
3     uint32_t      saddr;           /*      0      4 */
4     uint32_t      daddr;           /*      4      4 */
5     uint16_t      sport;           /*      8      2 */
6     uint16_t      dport;           /*     10      2 */
7     uint8_t       state;           /*     12      1 */
8     uint8_t       protocol;        /*     13      1 */
9     uint64_t      bytes_rx;        /*     16      8 */
10    uint64_t      packets_rx;      /*     24      8 */
11    uint32_t      rx_queue;        /*     32      4 */
12    uint32_t      last_ack;        /*     60      4 */
13    /* --- cacheline 1 boundary (64 bytes) --- */
14    ...
15    /* size: 192, cachelines: 3, members: 26 */
16 } __attribute__((__aligned__(64)));
```

- Where each member sits, cacheline by cacheline
- No idea which members are hot

PAHOLE + JSON: THE OVERLAY

```
1 $ perf report -s type --data-type-json=profile.json
2 $ pahole --perf-data-type=profile.json -C net_conn ~/bin/perf
3 struct net_conn {
4     uint32_t      saddr;           /*      0      4 |    8.7% R:3.1K */
5     uint32_t      daddr;           /*      4      4 |    9.1% R:3.2K */
6     ...
7     uint64_t      bytes_rx;        /*     16      8 |    7.6% R:33 W:2.7K
8     uint64_t      packets_rx;      /*     24      8 |    0.4% R:19 W:134
9     uint32_t      rx_queue;        /*     32      4 |    1.5% W:537 */
10    uint32_t      last_ack;         /*     60      4 |    1.1% W:379 */
11    ...
```

- Loads (R) and stores (W) per member
- Per cacheline summary: counts and periods

PAHOLE + JSON: THE OVERLAY

```
1 $ perf report -s type --data-type-json=profile.json
2 $ pahole --perf-data-type=profile.json -C net_conn ~/bin/perf
3 struct net_conn {
4     uint32_t      saddr;           /*      0      4 |    8.7% R:3.1K */
5     uint32_t      daddr;           /*      4      4 |    9.1% R:3.2K */
6     ...
7     uint64_t      bytes_rx;        /*     16      8 |    7.6% R:33 W:2.7K
8     uint64_t      packets_rx;     /*     24      8 |    0.4% R:19 W:134
9     uint32_t      rx_queue;        /*     32      4 |    1.5% W:537 */
10    uint32_t      last_ack;        /*     60      4 |    1.1% W:379 */
11    ...
```

- Loads (R) and stores (W) per member
- Per cacheline summary: counts and periods

PAHOLE + JSON: THE OVERLAY

```
1 $ perf report -s type --data-type-json=profile.json
2 $ pahole --perf-data-type=profile.json -C net_conn ~/bin/perf
3 struct net_conn {
4     uint32_t      saddr;           /*      0      4 |    8.7% R:3.1K */
5     uint32_t      daddr;           /*      4      4 |    9.1% R:3.2K */
6     ...
7     uint64_t      bytes_rx;        /*     16      8 |    7.6% R:33 W:2.7K
8     uint64_t      packets_rx;      /*     24      8 |    0.4% R:19 W:134
9     uint32_t      rx_queue;        /*     32      4 |    1.5% W:537 */
10    uint32_t      last_ack;         /*     60      4 |    1.1% W:379 */
11    ...
```

- Loads (R) and stores (W) per member
- Per cacheline summary: counts and periods

PAHOLE + JSON: THE OVERLAY

```
1 $ perf report -s type --data-type-json=profile.json
2 $ pahole --perf-data-type=profile.json -C net_conn ~/bin/perf
3 struct net_conn {
4     uint32_t      saddr;           /*      0      4 |    8.7% R:3.1K */
5     uint32_t      daddr;           /*      4      4 |    9.1% R:3.2K */
6     ...
7     uint64_t      bytes_rx;        /*     16     8 |    7.6% R:33 W:2.7K
8     uint64_t      packets_rx;      /*     24     8 |    0.4% R:19 W:134
9     uint32_t      rx_queue;        /*     32     4 |    1.5% W:537 */
10    uint32_t      last_ack;         /*     60     4 |    1.1% W:379 */
11    ...
```

- Loads (R) and stores (W) per member
- Per cacheline summary: counts and periods

PAHOLE + CTF: FALSE SHARING

```
1 $ perf data convert --to-ctf=./ctf.dir --data-type -i perf.data
2 $ pahole -F dwarf --perf-data-type=./ctf.dir -C net_conn ~/bin/perf
3 struct net_conn {
4     ...
5     >>> FALSE SHARING: cacheline 0: saddr / bytes_rx
6     ...
7 } __attribute__((__aligned__(64)));
```

- Each CTF sample keeps CPU, time, data address
- Same instance, other CPU, a write, within 10us
- Per-member counts match the JSON path

PAHOLE + CTF: FALSE SHARING

```
1 $ perf data convert --to-ctf=./ctf.dir --data-type -i perf.data
2 $ pahole -F dwarf --perf-data-type=./ctf.dir -C net_conn ~/bin/perf
3 struct net_conn {
4     ...
5     >>> FALSE SHARING: cacheline 0: saddr / bytes_rx
6     ...
7 } __attribute__((__aligned__(64)));
```

- Each CTF sample keeps CPU, time, data address
- Same instance, other CPU, a write, within 10us
- Per-member counts match the JSON path

PAHOLE + CTF: FALSE SHARING

```
1 $ perf data convert --to-ctf=./ctf.dir --data-type -i perf.data
2 $ pahole -F dwarf --perf-data-type=./ctf.dir -C net_conn ~/bin/perf
3 struct net_conn {
4     ...
5     >>> FALSE SHARING: cacheline 0: saddr / bytes_rx
6     ...
7 } __attribute__((__aligned__(64)));
```

- Each CTF sample keeps CPU, time, data address
- Same instance, other CPU, a write, within 10us
- Per-member counts match the JSON path

WHAT COMPILERS COULD GIVE US

- perf rebuilds types: DWARF + [disassembly](#)
- Coverage has a ceiling: unknowns remain
- GCC keeps the field at final: [MEM_EXPR](#)
- Ask: access records in the binary, like [CO-RE](#)
- Full story: [GNU Tools Cauldron talk](#)

LESSONS, AND WHAT IS NEXT

- Compilers are inconsistent: read the DWARF
- Tests must be environment-aware
- Dumps save hours of debugging
- Rust: niche layouts, const generics, async fn
- [DWARF 6](#): new tags, adoption varies
- BTF extensions for what C can't express

BTF: RECENT DEVELOPMENTS

- Alan Maguire (Oracle), co-maintainer
- Inline sites
- True signatures
- Rust and other languages
- Rust types in kernel BTF: an experiment
- BTF self-description

INLINE SITES

```
1 $ pahole -J --btf_features=default,inline \  
2     --btf_encode_detached=/tmp/vmlinux+inline vmlinux  
3 $ pfunct --inline_sites --elf=vmlinux /tmp/vmlinux+inline  
4 0xffffffff8120005f [trace_initcall_start_cb+0x1f, ...] get_current(void)  
5 0xffffffff81200097 [trace_initcall_level+0x7, ...]  
6     __do_trace_initcall_level(const char * level [%rdi])
```

- Where each inline lands, with its parameters
- New kinds: LOC_PROTO, PARAM, LOCSEC
- Opt-in: --btf_features=default,inline
- v3 in review, needs libbpf location APIs

INLINE SITES

```
1 $ pahole -J --btf_features=default,inline \  
2   --btf_encode_detached=/tmp/vmlinux+inline vmlinux  
3 $ pfunct --inline_sites --elf=vmlinux /tmp/vmlinux+inline  
4 0xffffffff8120005f [trace_initcall_start_cb+0x1f, ...] get_current(void)  
5 0xffffffff81200097 [trace_initcall_level+0x7, ...]  
6   __do_trace_initcall_level(const char * level [%rdi])
```

- Where each inline lands, with its parameters
- New kinds: LOC_PROTO, PARAM, LOCSEC
- Opt-in: --btf_features=default,inline
- v3 in review, needs libbpf location APIs

INLINE SITES

```
1 $ pahole -J --btf_features=default,inline \  
2     --btf_encode_detached=/tmp/vmlinux+inline vmlinux  
3 $ pfunct --inline_sites --elf=vmlinux /tmp/vmlinux+inline  
4 0xffffffff8120005f [trace_initcall_start_cb+0x1f, ...] get_current(void)  
5 0xffffffff81200097 [trace_initcall_level+0x7, ...]  
6     __do_trace_initcall_level(const char * level [%rdi])
```

- Where each inline lands, with its parameters
- New kinds: LOC_PROTO, PARAM, LOCSEC
- Opt-in: --btf_features=default,inline
- v3 in review, needs libbpf location APIs

TRUE SIGNATURES

```
1 /* source */
2 static int map_create(union bpf_attr *attr, bpfptr_t uattr);
3 /* as compiled */
4 static int map_create(union bpf_attr *attr, bool uattr__is_kernel);
```

- BTF had source signatures; compilers change them
- Mostly dead argument elimination
- Kept only if DWARF locations match the ABI
- x86_64: 875 of 69103 changed, 20 left out
- [Yonghong Song, pahole 1.32](#): +true_signature

TRUE SIGNATURES

```
1 /* source */
2 static int map_create(union bpf_attr *attr, bpfptr_t uattr);
3 /* as compiled */
4 static int map_create(union bpf_attr *attr, bool uattr__is_kernel);
```

- BTF had source signatures; compilers change them
- Mostly dead argument elimination
- Kept only if DWARF locations match the ABI
- x86_64: 875 of 69103 changed, 20 left out
- [Yonghong Song, pahole 1.32](#): +true_signature

TRUE SIGNATURES

```
1 /* source */
2 static int map_create(union bpf_attr *attr, bpfptr_t uattr);
3 /* as compiled */
4 static int map_create(union bpf_attr *attr, bool uattr__is_kernel);
```

- BTF had source signatures; compilers change them
- Mostly dead argument elimination
- Kept only if DWARF locations match the ABI
- x86_64: 875 of 69103 changed, 20 left out
- [Yonghong Song, pahole 1.32](#): +true_signature

RUST AND OTHER LANGUAGES

```
1 struct Option<u32> {  
2     u32 __discriminant;      /* 0 4 */  
3     union {  
4         struct None {};  
5         struct Some { u32 __0; /* 4 4 */ };  
6     };  
7 };
```

- Rust enums: tagged unions, DW_TAG_variant_part
- __discriminant from DW_AT_discr, not inferred
- Mangled names: DECL_TAG, trace by plain name
- Details: Extending BTF, eBPF track

RUST AND OTHER LANGUAGES

```
1 struct Option<u32> {  
2     u32 __discriminant;      /* 0 4 */  
3     union {  
4         struct None {};  
5         struct Some { u32 __0; /* 4 4 */ };  
6     };  
7 };
```

- Rust enums: tagged unions, DW_TAG_variant_part
- __discriminant from DW_AT_discr, not inferred
- Mangled names: [DECL_TAG](#), trace by plain name
- Details: [Extending BTF](#), [eBPF track](#)

RUST AND OTHER LANGUAGES

```
1 struct Option<u32> {  
2     u32 __discriminant;      /* 0 4 */  
3     union {  
4         struct None {};  
5         struct Some { u32 __0; /* 4 4 */ };  
6     };  
7 };
```

- Rust enums: tagged unions, DW_TAG_variant_part
- __discriminant from DW_AT_discr, not inferred
- Mangled names: [DECL_TAG](#), trace by plain name
- Details: [Extending BTF](#), [eBPF track](#)

EXPERIMENTING WITH MODULE BTF GENERATION IN RUST

- remove lang-exclude for Rust CUs
- make LLVM=1
- With latest pahole v1.32, it works!
- We see many duplicate types in vmlinux BTF due to bindgen differences (bpf_link, sock)

C TYPE

```
struct leaf_0x2_table {
    union {
        enum _cache_table_type c_type;
        enum _tlb_table_type t_type;
    };
    /* offset 0, size 1 */
    /* 1 byte of padding */
    union {
        short c_size;
        short entries;
    };
    /* offset 2, size 2 */
};
```

RUST TYPE

```
struct leaf_0x2_table_bindgen {  
    union {  
        uint8_t c_type;  
        uint8_t t_type;  
    } __bindgen_anon_1;           /* offset 0, size 1 */  
    /* 1 byte of padding */  
    union {  
        int16_t c_size;  
        int16_t entries;  
    } __bindgen_anon_2;         /* offset 2, size 2 */  
};
```

AND ON REBOOT?

- vmlinux BTF load fails due to long names
- If we increase name limit to 2048 bytes, all works!
- Can load rust samples with BTF

BTF SELF-DESCRIPTION

- Kind layout: BTF describes its own kinds
- Parse BTF even with kinds you don't know
- Makes life easier for tools using BTF

THE END

- <https://github.com/acmel/dwarves>
- acme@redhat.com
- alan.maguire@oracle.com
- dwarves@vger.kernel.org
- <https://bsky.app/profile/acmel.bsky.social>



fedorapeople.org/~acmel/
lpc-2026-pahole-2025

ECOSYSTEM, NOT ALONE

- `libabigail` abidiff: full ABI compat
- `codiff`: lighter dump diff
- `gdb ptype /o` (8.1): offsets in the debugger
- `dwgrep`: query DWARF DIEs
- `DHAT`: heap access histograms
- Compilers emit BTF now (`GCC -gbtf`, `LLVM -gbtf`)
- `pahole` still dedups/repairs

VMLINUX BTF AS A MODULE

- Small VMs: ~5.4 MB of BTF, mostly unused
- **v4**: CONFIG_DEBUG_INFO_BTF=m, loaded on first use
- Pushback: ~2k lines for maybe-saved memory
- Alternative: **zstd in the image**, 6.7 MB → 2.2 MB
- Alexei: compression the only one worth considering
- Inline BTF: a module is still the plan (.BTF.link)

OLD TOOLS, NEW BTF

- AlmaLinux 8 containers: [bpftool v4.18.0](#)
- No .bss DATASEC, no VAR in its dumps
- bpftool v7.5+: correct
- Not a pahole bug: skip, and say why

```
if echo "$dump" | grep -q "DATASEC '\.bss'"; then
    check_bss_variable
else
    info_log "skip: bpftool doesn't display .bss DATASEC"
fi
```

WHEN REPLAY IS NEEDED

- Pointer chasing: foo->bar->baz
- Call results: %rax from the return type
- Copies and spills: mov, stack slots
- Gaps: variable optimized out at this PC
- Pointer arithmetic: lea, add, container_of()
- percpu, %gs and the stack canary