LinuxCon + ContainerCon + CloudOpen China 2018

eBPF in Action

李枫
hkli2013@126.com
Jun 25，2018

Agenda

I. Anatomy of eBPF

- Overview
- Internal
- LLVM
- Development
- BCC
- Languages
- Application
- Pros & Cons

II. XDP

- Overview
- AF_XDP
- Hardware Offloading

III. eBPF-driven Secure Network

- Cilium
- Suricata
- Interaction with Go
- bpfilter

IV. eBPF on ARM

- ARM Development Boards
- Distributions for AARCH64
- My Practice

V. Wrap-up

I. Anatomy of eBPF

1) Overview

- https://en.wikipedia.org/wiki/Berkeley_Packet_Filter

BPF (Berkeley Packet Filter, aka cBPF)

- Introduced in kernel 2.1.75 (1997)
- https://blog.cloudflare.com/bpf-the-forgotten-bytecode/
-

```
# tcpdump host 127.0.0.1 and port 22 -d
(000) ldh      [12]
(001) jeq      #0x800          jt 2    jf 18
(002) ld       [26]
(003) jeq      #0x7f000001     jt 6    jf 4
(004) ld       [30]
(005) jeq      #0x7f000001     jt 6    jf 18
(006) ldb      [23]
(007) jeq      #0x84           jt 10   jf 8
(008) jeq      #0x6            jt 10   jf 9
(009) jeq      #0x11           jt 10   jf 18
(010) ldh      [20]
(011) jset     #0x1fff         jt 18   jf 12
(012) ldxb     4*([14]&0xf)
(013) ldh      [x + 14]
[...]
```

Optimizes packet filter performance

2 x 32-bit registers & scratch memory

User-defined bytecode executed by an in-kernel sandboxed virtual machine

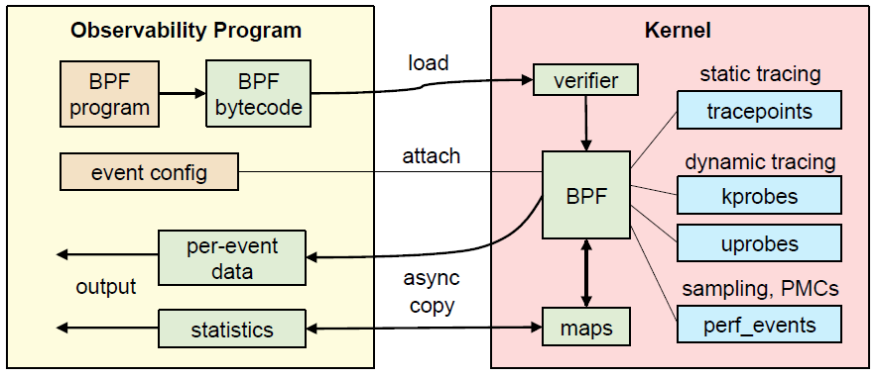
Steven McCanne and Van Jacobson, 1993

Source: https://www.slideshare.net/brendangregg/kernel-recipes-2017-performance-analysis-with-bpf

eBPF (extended BPF)

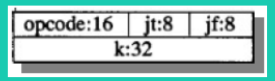
- Since Linux Kernel v3.15 and ongoing
- aims at being **a universal in-kernel virtual machine**
- **a simple way to extend the functionality of Kernel at runtime**
- **“dtrace for Linux”**

changing the old ways for Kernel instrumentation



Source: https://www.slideshare.net/brendangregg/kernel-recipes-2017-performance-analysis-with-bpf

Comparison

	cBPF	eBPF
Register	**Two 32 bit registers:** A: accumulator X: indexing	**Eleven 64 bit registers:** R0: return value/exit value R1-R5: arguments R6-R9: callee saved registers R10: read-only frame pointer
Instruction	~30 	~90 
JIT	Support	Support (better mapping with newer architectures for JITing)
Toolchain	GCC, tools/net	LLVM eBPF backend
Platform	x86_64, ARM, ARM64, SPARC, PowerPC, MIPS and s390	x86-64, aarch64, s390x...
System Call		 (CALL, MAP, LOAD...)
Application	tcpdump... apply for seccomp filters, traffic control…	DDoS Mitigation, Intrusion Detection, Container Security, SDN Configuration, Observability...

opcode:16	jt:8	jf:8		
k:32				

op:8	dst:4	src:4	off:16
imm:32			

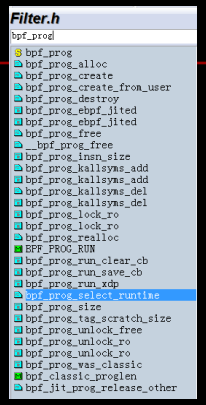
```
#include <linux/bpf.h>

int bpf(int cmd, union bpf_attr *attr, unsigned int size);
```

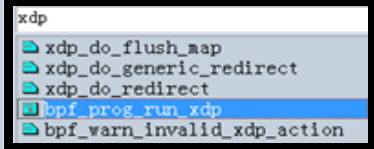
2) Internal (base on 4.18-rc1)

- $KERNEL_SRC/Documentation/networking/filter.txt
- $KERNEL_SRC/include/linux/filter.h

```c
/* Helper macros for filter block array initializers. */

/* ALU ops on registers, bpf_add|sub|...: dst_reg += src_reg */

#define BPF_ALU64_REG(OP, DST, SRC)                 \
    ((struct bpf_insn) {                        \
        .code  = BPF_ALU64 | BPF_OP(OP) | BPF_X,    \
        .dst_reg = DST,                     \
        .src_reg = SRC,                     \
        .off   = 0,                     \
        .imm   = 0 })

#define BPF_ALU32_REG(OP, DST, SRC)                 \
    ((struct bpf_insn) {                        \
        .code  = BPF_ALU | BPF_OP(OP) | BPF_X,      \
        .dst_reg = DST,                     \
        .src_reg = SRC,                     \
        .off   = 0,                     \
        .imm   = 0 })
```

```c
.insns = {
    BPF_MOV64_REG(BPF_REG_2, BPF_REG_10),
    BPF_ALU64_IMM(BPF_ADD, BPF_REG_2, -8),
    BPF_ST_MEM(BPF_DW, BPF_REG_2, 0, 0),
    BPF_LD_MAP_FD(BPF_REG_1, 0),
    BPF_EMIT_CALL(BPF_FUNC_map_lookup_elem),
    BPF_MOV64_REG(BPF_REG_1, BPF_REG_10),
    BPF_ALU64_IMM(BPF_ADD, BPF_REG_1, -152),
    BPF_STX_MEM(BPF_DW, BPF_REG_1, BPF_REG_0, 0),
    BPF_JMP_IMM(BPF_JEQ, BPF_REG_0, 0, 2),
    BPF_LDX_MEM(BPF_DW, BPF_REG_3, BPF_REG_1, 0),
    BPF_ST_MEM(BPF_DW, BPF_REG_3, 0, 42),
    BPF_EXIT_INSN(),
}
```






```c
static __always_inline u32 bpf_prog_run_xdp(const struct bpf_prog *prog,
                             struct xdp_buff *xdp)
{
    /* Caller needs to hold rcu_read_lock() (!), otherwise program
     * can be released while still running, or map elements could be
     * freed early while still having concurrent users. XDP fastpath
     * already takes rcu_read_lock() when fetching the program, so
     * it's not necessary here anymore.
     */
    return BPF_PROG_RUN(prog, xdp);
}
```

```c
#define BPF_PROG_RUN(filter, ctx)  (*(filter)->bpf_func)(ctx, (filter)->insnsi)
```

```c
struct bpf_prog {
    u16             pages;      /* Number of allocated pages */
    u16             jited:1,    /* Is our filter JIT'ed? */
                    jit_requested:1,/* archs need to JIT the prog */
                    locked:1,   /* Program image locked? */
                    gpl_compatible:1, /* Is filter GPL compatible? */
                    cb_access:1,    /* Is control block accessed? */
                    dst_needed:1,   /* Do we need dst entry? */
                    blinded:1,  /* Was blinded */
                    is_func:1,  /* program is a bpf function */
                    kprobe_override:1, /* Do we override a kprobe? */
                    has_callchain_buf:1; /* callchain buffer allocated? */
    enum bpf_prog_type  type;       /* Type of BPF program */
    enum bpf_attach_type    expected_attach_type; /* For some prog types */
    u32             len;        /* Number of filter blocks */
    u32             jited_len;  /* Size of jited insns in bytes */
    u8              tag[BPF_TAG_SIZE];
    struct bpf_prog_aux *aux;       /* Auxiliary fields */
    struct sock_fprog_kern  *orig_prog; /* Original BPF program */
    unsigned int        (*bpf_func)(const void *ctx,
                        const struct bpf_insn *insn);
    /* Instructions for interpreter */
    union {
        struct sock_filter  insns[0];
        struct bpf_insn     insnsi[0];
    };
} ? end bpf_prog ? ;
```

- https://github.com/iovisor/bpf-docs/blob/master/eBPF.md

$KERNEL_SRC/include/uapi/linux/bpf.h

```c
enum bpf_prog_type {
    BPF_PROG_TYPE_UNSPEC,
    BPF_PROG_TYPE_SOCKET_FILTER,
    BPF_PROG_TYPE_KPROBE,
    BPF_PROG_TYPE_SCHED_CLS,
    BPF_PROG_TYPE_SCHED_ACT,
    BPF_PROG_TYPE_TRACEPOINT,
    BPF_PROG_TYPE_XDP,
    BPF_PROG_TYPE_PERF_EVENT,
    BPF_PROG_TYPE_CGROUP_SKB,
    BPF_PROG_TYPE_CGROUP_SOCK,
    BPF_PROG_TYPE_LWT_IN,
    BPF_PROG_TYPE_LWT_OUT,
    BPF_PROG_TYPE_LWT_XMIT,
    BPF_PROG_TYPE_SOCK_OPS,
    BPF_PROG_TYPE_SK_SKB,
    BPF_PROG_TYPE_CGROUP_DEVICE,
    BPF_PROG_TYPE_SK_MSG,
    BPF_PROG_TYPE_RAW_TRACEPOINT,
    BPF_PROG_TYPE_CGROUP_SOCK_ADDR,
    BPF_PROG_TYPE_LWT_SEG6LOCAL,
    BPF_PROG_TYPE_LIRC_MODE2,
};
```

```c
struct bpf_insn {
    __u8    code;       /* opcode */
    __u8    dst_reg:4;  /* dest register */
    __u8    src_reg:4;  /* source register */
    __s16   off;        /* signed offset */
    __s32   imm;        /* signed immediate constant */
};
```

```c
enum bpf_map_type {
    BPF_MAP_TYPE_UNSPEC,
    BPF_MAP_TYPE_HASH,
    BPF_MAP_TYPE_ARRAY,
    BPF_MAP_TYPE_PROG_ARRAY,
    BPF_MAP_TYPE_PERF_EVENT_ARRAY,
    BPF_MAP_TYPE_PERCPU_HASH,
    BPF_MAP_TYPE_PERCPU_ARRAY,
    BPF_MAP_TYPE_STACK_TRACE,
    BPF_MAP_TYPE_CGROUP_ARRAY,
    BPF_MAP_TYPE_LRU_HASH,
    BPF_MAP_TYPE_LRU_PERCPU_HASH,
    BPF_MAP_TYPE_LPM_TRIE,
    BPF_MAP_TYPE_ARRAY_OF_MAPS,
    BPF_MAP_TYPE_HASH_OF_MAPS,
    BPF_MAP_TYPE_DEVMAP,
    BPF_MAP_TYPE_SOCKMAP,
    BPF_MAP_TYPE_CPUMAP,
    BPF_MAP_TYPE_XSKMAP,
    BPF_MAP_TYPE_SOCKHASH,
};
```

```c
/* User return codes for XDP prog type.
 * A valid XDP program must return one of these defined values. All other
 * return codes are reserved for future use. Unknown return codes will
 * result in packet drops and a warning via bpf_warn_invalid_xdp_action().
 */
enum xdp_action {
    XDP_ABORTED = 0,
    XDP_DROP,
    XDP_PASS,
    XDP_TX,
    XDP_REDIRECT,
};

/* user accessible metadata for XDP packet hook
 * new fields must be added to the end of this structure
 */
struct xdp_md {
    __u32 data;
    __u32 data_end;
    __u32 data_meta;
    /* Below access go through struct xdp_rxq_info */
    __u32 ingress_ifindex; /* rxq->dev->ifindex */
    __u32 rx_queue_index;  /* rxq->queue_index  */
};
```

```c
#define __BPF_FUNC_MAPPER(FN)       \
    FN(unspec),         \
    FN(map_lookup_elem),        \
    FN(map_update_elem),        \
    FN(map_delete_elem),        \
    FN(probe_read),         \
    FN(ktime_get_ns),       \
    FN(trace_printk),       \
    FN(get_prandom_u32),        \
    FN(get_smp_processor_id),   \
    FN(skb_store_bytes),        \
    FN(l3_csum_replace),        \
    FN(l4_csum_replace),        \
    FN(tail_call),          \
    FN(clone_redirect),     \
    FN(get_current_pid_tgid),   \
    FN(get_current_uid_gid),    \
```

...

```c
struct bpf_prog_info {
    __u32 type;
    __u32 id;
    __u8  tag[BPF_TAG_SIZE];
    __u32 jited_prog_len;
    __u32 xlated_prog_len;
    __aligned_u64 jited_prog_insns;
    __aligned_u64 xlated_prog_insns;
    __u64 load_time;    /* ns since boottime */
    __u32 created_by_uid;
    __u32 nr_map_ids;
    __aligned_u64 map_ids;
    char name[BPF_OBJ_NAME_LEN];
    __u32 ifindex;
    __u32 gpl_compatible:1;
    __u64 netns_dev;
    __u64 netns_ino;
    __u32 nr_jited_ksyms;
    __u32 nr_jited_func_lens;
    __aligned_u64 jited_ksyms;
    __aligned_u64 jited_func_lens;
} __attribute__((aligned(8)));
```

```c
struct bpf_map_info {
    __u32 type;
    __u32 id;
    __u32 key_size;
    __u32 value_size;
    __u32 max_entries;
    __u32 map_flags;
    char  name[BPF_OBJ_NAME_LEN];
    __u32 ifindex;
    __u32 :32;
    __u64 netns_dev;
    __u64 netns_ino;
    __u32 btf_id;
    __u32 btf_key_type_id;
    __u32 btf_value_type_id;
} __attribute__((aligned(8)));
```

```c
union bpf_attr {
    ...
```

```c
/* integer value in 'imm' field of BPF_CALL instruction selects which helper
 * function eBPF program intends to call
 */
#define __BPF_ENUM_FN(x) BPF_FUNC_ ## x
enum bpf_func_id {
    __BPF_FUNC_MAPPER(__BPF_ENUM_FN)
    __BPF_FUNC_MAX_ID,
};
#undef __BPF_ENUM_FN
```

- **$KERNEL_SRC/kernel/bpf**

```
kernel/bpf
├── arraymap.c
├── bpf_lru_list.c
├── bpf_lru_list.h
├── cgroup.c
├── core.c
├── cpumap.c
├── devmap.c
├── disasm.c
├── disasm.h
├── hashtab.c
├── helpers.c
├── inode.c
├── lpm_trie.c
├── Makefile
├── map_in_map.c
├── map_in_map.h
├── offload.c
├── percpu_freelist.c
├── percpu_freelist.h
├── sockmap.c
├── stackmap.c
├── syscall.c
├── tnum.c
└── verifier.c
```

```
struct bpf_prog *bpf_prog_select_runtime(struct bpf_prog *fp, int *err)
```

```
 * Try to JIT eBPF program, if JIT is not available, use interpreter.
 * The BPF program will be executed via BPF_PROG_RUN() macro.
 */
struct bpf_prog *bpf_prog_select_runtime(struct bpf_prog *fp, int *err)
{
#ifndef CONFIG_BPF_JIT_ALWAYS_ON
    u32 stack_depth = max_t(u32, fp->aux->stack_depth, 1);

    fp->bpf_func = interpreters[(round_up(stack_depth, 32) / 32) - 1];
#else
    fp->bpf_func = __bpf_prog_ret0_warn;
#endif

    /* eBPF JITs can rewrite the program in case constant
     * blinding is active. However, in case of error during
     * blinding, bpf_int_jit_compile() must always return a
     * valid program, which in this case would simply not
     * be JITed, but falls back to the interpreter.
     */
    if (!bpf_prog_is_dev_bound(fp->aux)) {
        fp = bpf_int_jit_compile(fp);
#ifdef CONFIG_BPF_JIT_ALWAYS_ON
        if (!fp->jited) {
            *err = -ENOTSUPP;
            return fp;
        }
#endif
    } else {
        *err = bpf_prog_offload_compile(fp);
        if (*err)
            return fp;
    }
```

```
SYSCALL_DEFINE3(bpf, int, cmd, union bpf_attr __user *, uattr, unsigned int, size)
{
```

...

```
static int bpf_prog_load(union bpf_attr *attr)
```

```
int bpf_check(struct bpf_prog **prog, union bpf_attr *attr)
```

- **$KERNEL_SRC/arch/$ARCH/net/bpf_jit_compXX.c**
  $KERNEL_SRC/arch/$ARCH/net/ebpf_jit.c

```
struct bpf_prog *bpf_int_jit_compile(struct bpf_prog *prog)
```

3) LLVM

- eBPF backend firstly introduced in LLVM 3.7 release
- http:///llvm.org/docs/CodeGenerator.html#the-extended-berkeley-packet-filter-ebpf-backend
-
  - Enabled by default with all major distributions
    - Registered targets: `llc --version`
    - `llc`'s BPF `-march` options: bpf, bpfeb, bpfel
    - `llc`'s BPF `-mcpu` options: generic, v1, v2, probe
  - Assembler output through `-S` supported
  - `llvm-objdump` for disassembler and code annotations (via DWARF)
  - Annotations correlate directly with kernel verifier log
  - Outputs ELF file with maps as relocation entries
    - Processed by BPF loaders (e.g. iproute2) and pushed into kernel

Source: https://ossna2017.sched.com/event/BCsg/making-the-kernels-networking-data-path-programmable-with-bpf-and-xdp-daniel-borkmann-covalent

- **$LLVM_SRC/lib/Target/BPF**

```
BPF
├── AsmParser
│   ├── BPFAsmParser.cpp
│   ├── CMakeLists.txt
│   └── LLVMBuild.txt
├── BPFAsmPrinter.cpp
├── BPFCallingConv.td
├── BPFFrameLowering.cpp
├── BPFFrameLowering.h
├── BPF.h
├── BPFInstrFormats.td
├── BPFInstrInfo.cpp
├── BPFInstrInfo.h
├── BPFInstrInfo.td
├── BPFISelDAGToDAG.cpp
├── BPFISelLowering.cpp
├── BPFISelLowering.h
├── BPFMCInstLower.cpp
├── BPFMCInstLower.h
├── BPFMIPeephole.cpp
├── BPFRegisterInfo.cpp
├── BPFRegisterInfo.h
├── BPFRegisterInfo.td
├── BPFSubtarget.cpp
├── BPFSubtarget.h
├── BPFTargetMachine.cpp
├── BPFTargetMachine.h
├── BPF.td
├── CMakeLists.txt
├── Disassembler
│   ├── BPFDisassembler.cpp
│   ├── CMakeLists.txt
│   └── LLVMBuild.txt
├── InstPrinter
│   ├── BPFInstPrinter.cpp
│   ├── BPFInstPrinter.h
│   ├── CMakeLists.txt
│   └── LLVMBuild.txt
├── LLVMBuild.txt
├── MCTargetDesc
│   ├── BPFAsmBackend.cpp
│   ├── BPFELFObjectWriter.cpp
│   ├── BPFMCAsmInfo.h
│   ├── BPFMCCodeEmitter.cpp
│   ├── BPFMCTargetDesc.cpp
│   ├── BPFMCTargetDesc.h
│   ├── CMakeLists.txt
│   └── LLVMBuild.txt
└── TargetInfo
    ├── BPFTargetInfo.cpp
    ├── CMakeLists.txt
    └── LLVMBuild.txt
```

```
[myrpi4@promote Target]$ cloc BPF
      46 text files.
      46 unique files.
      11 files ignored.

github.com/AlDanial/cloc v 1.72  T=1.69 s (20.7 files/s, 2628.5 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C++                             18            603            443           2715
C/C++ Header                    11            120            154            355
CMake                            6              5              0             46
-------------------------------------------------------------------------------
SUM:                            35            728            597           3116
-------------------------------------------------------------------------------
```

- http://cilium.readthedocs.io/en/latest/bpf/

LLVM vs GCC

- https://en.wikipedia.org/wiki/LLVM
- http://clang.llvm.org/

GCC	LLVM
GPL v3	UIUC, MIT
Front-end: CC1 / CPP	Front-end: Clang
ld.bfd / ld.gold	lld / mclinker
gdb	lldb
as / objdump	MC layer
libstdc++	libc++
libsupc++	libc++abi
libgcc	libcompiler-rt
libgccjit	libLLVMMCJIT
	ORC JIT, Coroutines, Clangd, libclc, ...

How is LLVM being used today?

XCode, Swift
FreeBSD, OpenMandriva Lx
Android
Debian experimenting with Clang as an additional compiler
...

Clang Goals

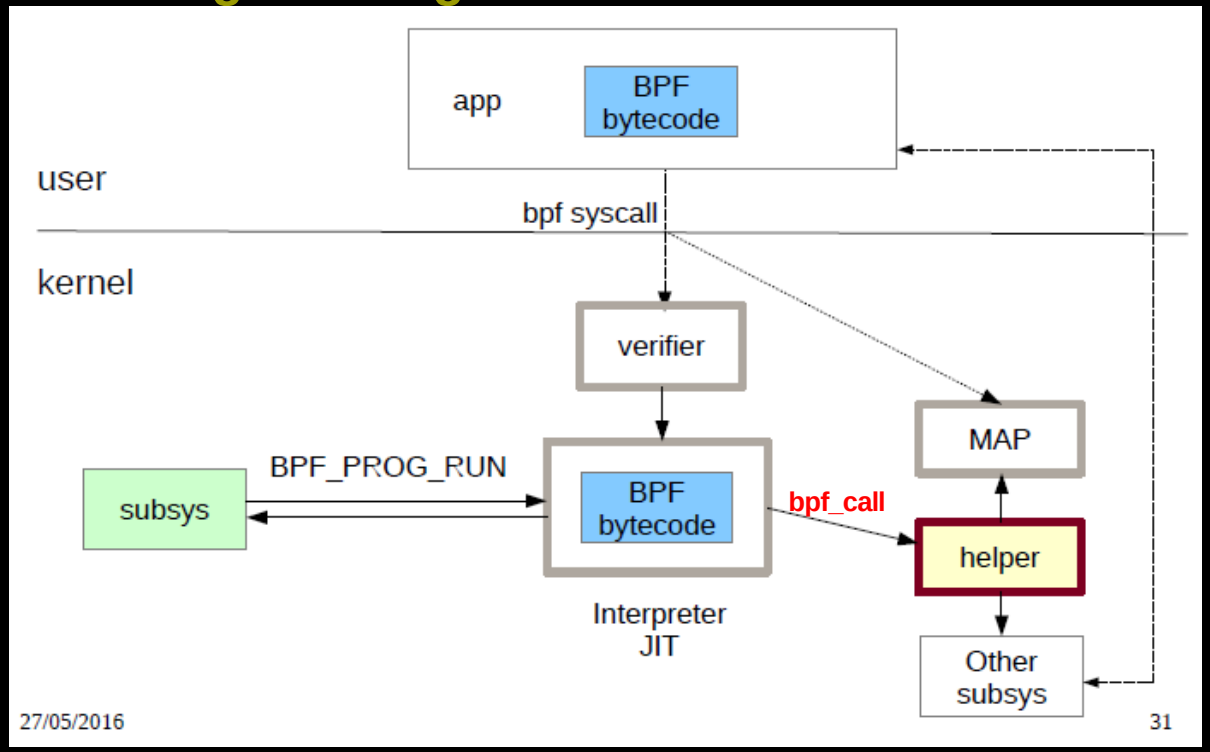
- GCC compatibility
- Fast compilation and low memory footprints
- Can reduce the linking time
- User friendly diagnostics
- Tooling
  - static analyzers
  - sanitizers

$KERNEL_SRC/samples/bpf/Makefile

4) Development

- Methods
  1) eBPF assembly
  2) BCC

  …
- BPF Programming Flow



Source: http://www.slideshare.net/vh21/meet-cutebetweenebpfandtracing

Debugging

- https://www.netronome.com/media/documents/UG_Getting_Started_with_eBPF_Offload.pdf
- int bpf(int cmd, union bpf_attr *attr, unsigned int size);

log_level

- 0: No debug output.
- 1: Debug information from the verifier (all instructions).
- 2: More information: add all register states after each instruction.

- llvm-objdump, llvm-mc…
- BCC
  $BCC_SRC/src/cc/bpf_module.h

```
// Options to enable different debug logging.
enum {
  // Debug output compiled LLVM IR.
  DEBUG_LLVM_IR = 0x1,
  // Debug output loaded BPF bytecode and register state on branches.
  DEBUG_BPF = 0x2,
  // Debug output pre-processor result.
  DEBUG_PREPROCESSOR = 0x4,
  // Debug output ASM instructions embedded with source.
  DEBUG_SOURCE = 0x8,
  // Debug output register state on all instructions in addition to DEBUG_BPF.
  DEBUG_BPF_REGISTER_STATE = 0x10,
};
```

bpf_trace_printk() //for C
BPF.trace_print() //for Python

bpftool

- $KERNEL_SRC/tools/bpf

```
[mydev@myfedora bpf]$ tree -L 1 bpftool
bpftool
├── bash-completion
├── cfg.c
├── cfg.h
├── cgroup.c
├── common.c
├── Documentation
├── jit_disasm.c
├── json_writer.c
├── json_writer.h
├── main.c
├── main.h
├── Makefile
├── map.c
├── map_perf_ring.c
├── perf.c
├── prog.c
├── xlated_dumper.c
└── xlated_dumper.h
```

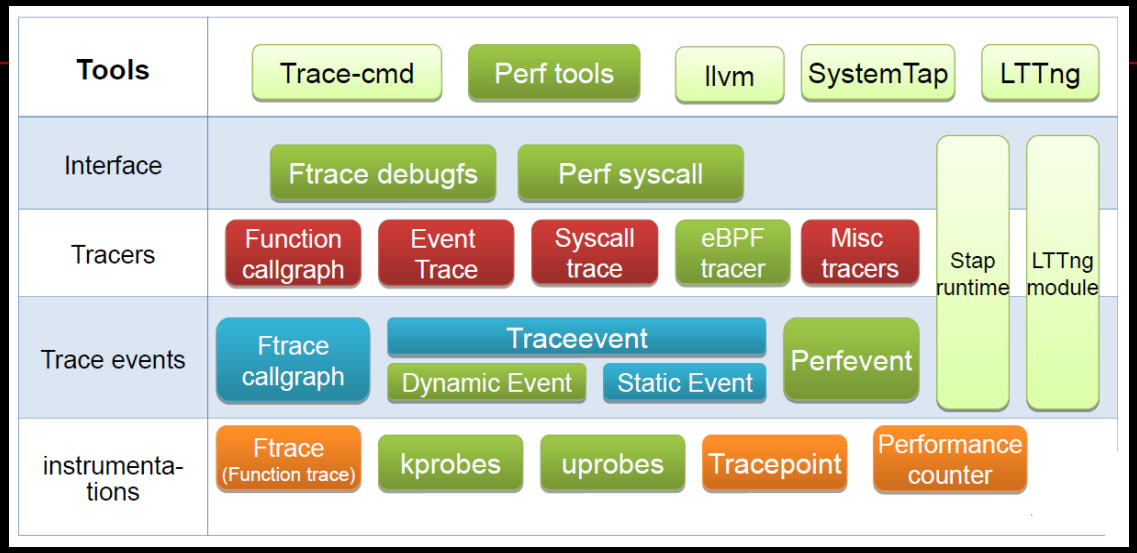
```
# bpftool prog show
1337: sched_cls  name cls_entry  tag e202124da7c84e89
      loaded_at Mar 08/19:53  uid 0
      xlated 304B  not jited  memlock 4096B
```

```
# bpftool prog dump xlated id 1337
   0: (71) r6 = *(u8 *)(r1 +142)
   1: (54) (u32) r6 &= (u32) 1
   2: (15) if r6 == 0x0 goto pc+7
   3: (bf) r6 = r1
   [...]
  37: (95) exit
```

```
# bpftool map
1234: array  name ch_rings  flags 0x0
      key 4B  value 4B  max_entries 7860  memlock 65536B

# bpftool map dump id 1234
key: 00 00 00 00  value: 00 00 00 00
key: 01 00 00 00  value: 00 00 00 00
key: 02 00 00 00  value: 00 00 00 00
key: 03 00 00 00  value: 00 00 00 00
[...]
Found 7860 elements
```

5) BCC (BPF Compiler Collection)

- https://iovisor.github.io/bcc/
- Kernel Tracing Landscape



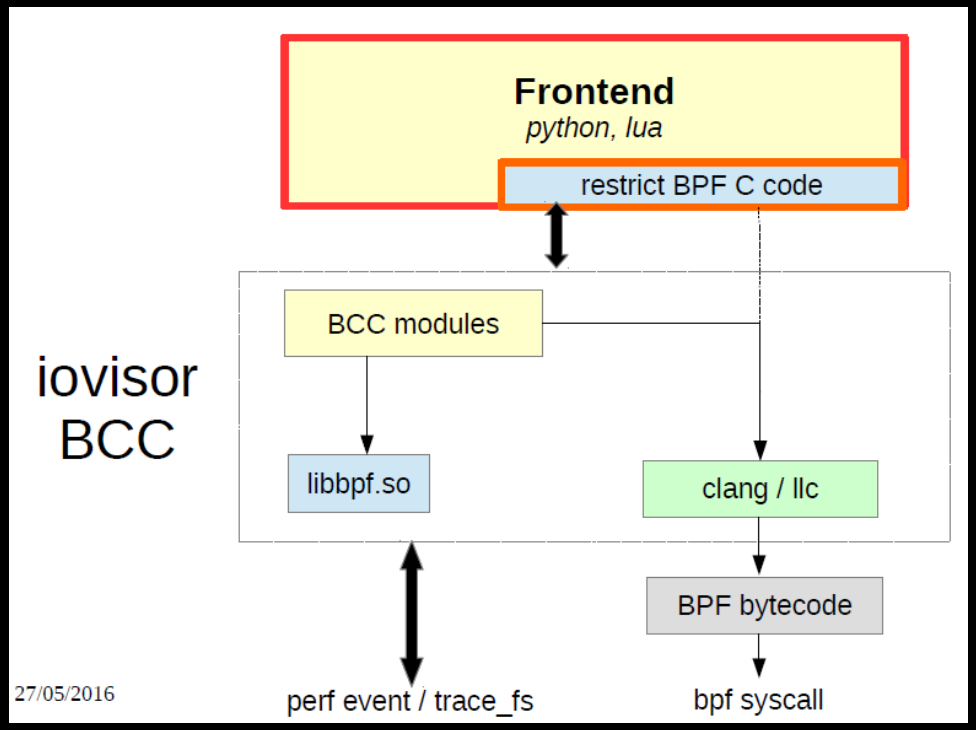
Source: http://tracingsummit.org/w/images/8/8c/TracingSummit2015-DynamicProbes.pdf

A toolkit with Python/Lua frontend for compiling, loading, and executing BPF programs, which allows user-defined instrumentation on a live kernel image:

- Compile BPF program from C source
- Attach BPF program to kprobe/uprobe/tracepoint/USDT/socket
- Poll data from BPF program

- Framework for building new tools or one-off scripts
- Contains a P4 compiler for BPF targets
- Additional projects to support Go, Rust, and DTrace-style frontend
- …

Arch

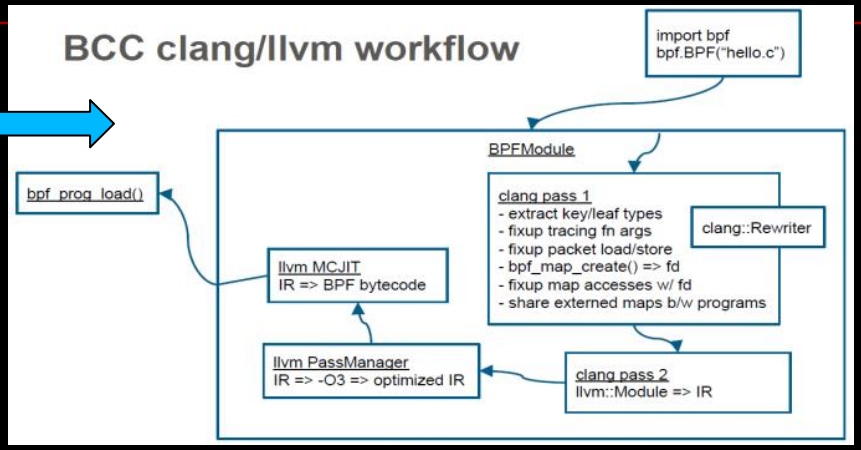
- 

Source: http://www.slideshare.net/vh21/meet-cutebetweenebpfandtracing

A Sample

- https://lwn.net/Articles/747640/ //Some advanced BCC topics

```
#!/usr/bin/env python

from bcc import BPF
from time import sleep

program = """
    BPF_HASH(callers, u64, unsigned long);

    TRACEPOINT_PROBE(kmem, kmalloc) {
        u64 ip = args->call_site;
        unsigned long *count;
        unsigned long c = 1;

        count = callers.lookup((u64 *)&ip);
        if (count != 0)
            c += *count;

        callers.update(&ip, &c);

        return 0;
    }
"""
b = BPF(text=program)

while True:
    try:
        sleep(1)
        for k,v in sorted(b["callers"].items()):
            print ("%s %u" % (b.ksym(k.value), v.value))
        print
    except KeyboardInterrupt:
        exit()
```



Source: http://linuxplumbersconf.org/2015/ocw//system/presentations/3249/original/bpf_llvm_2015aug19.pdf

- https://github.com/iovisor/bcc/blob/master/docs/tutorial.md
- https://github.com/iovisor/bcc/blob/master/docs/reference_guide.md
- https://github.com/iovisor/bcc/blob/master/docs/tutorial_bcc_python_developer.md

6) Languages

SystemTap

- https://developers.redhat.com/blog/2018/04/23/systemtaps-bpf-backend-tracepoint-support/

```
# stap --bpf example.stp
```

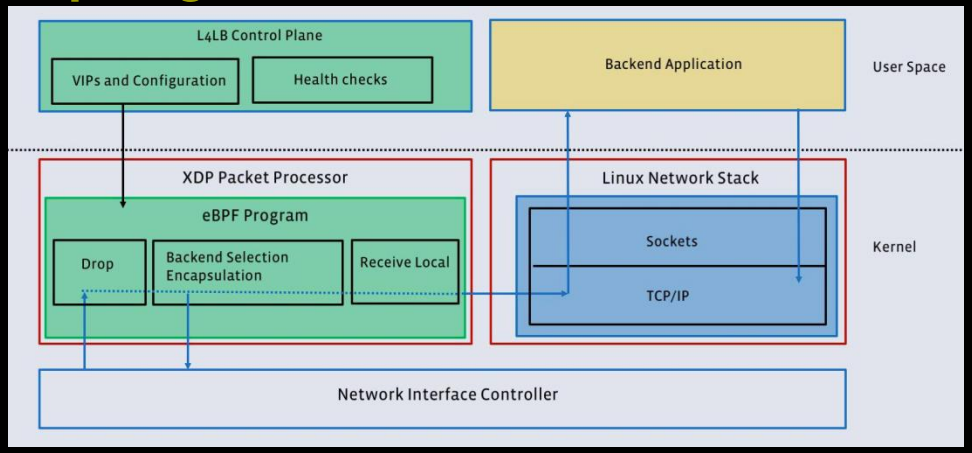
```
# stapbpf stap_1348.bo
```

- https://github.com/ajor/bpftrace

```
kprobe:[Ss]y[Ss]_*
{
  @[func] = count()
}
```

C++

- https://github.com/facebookincubator/katran



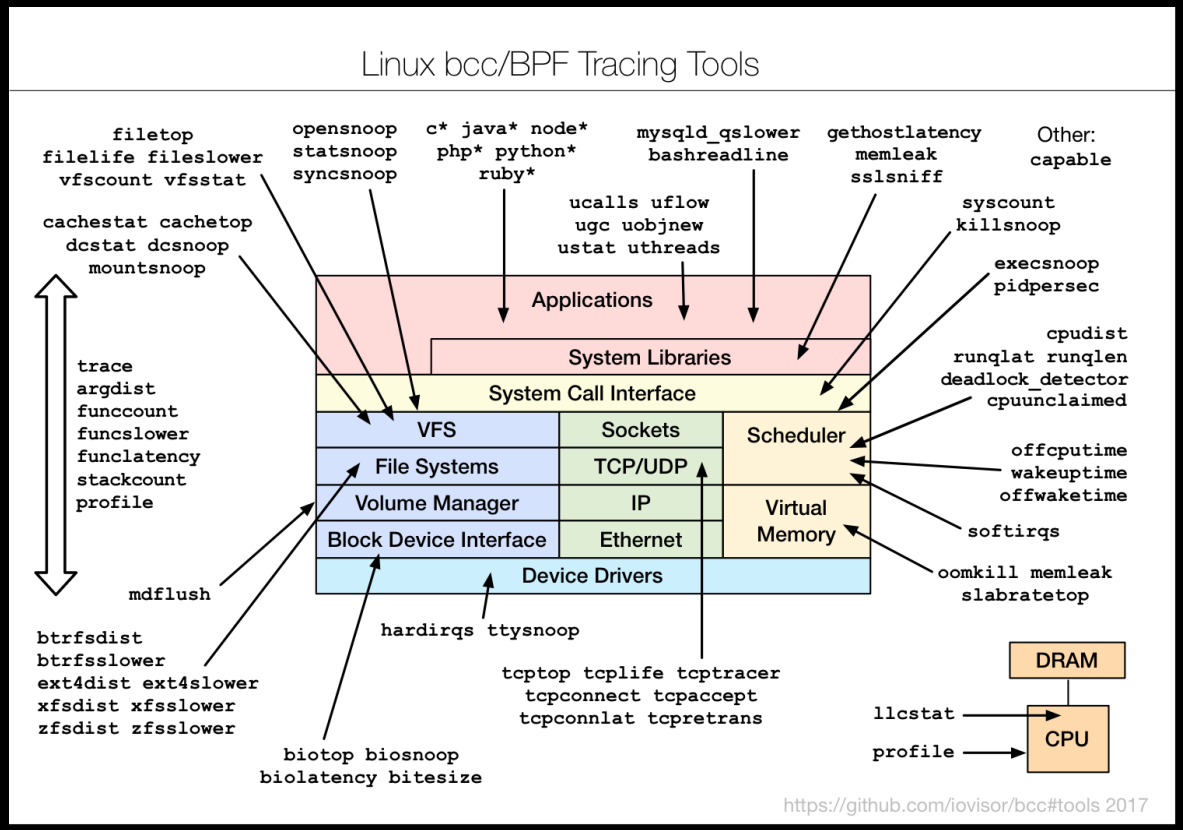
Source: http://www.infoq.com/cn/news/2018/06/Facebook-Katran

7) Applications

- http://cilium.readthedocs.io/en/latest/bpf/#projects-using-bpf

Tuning

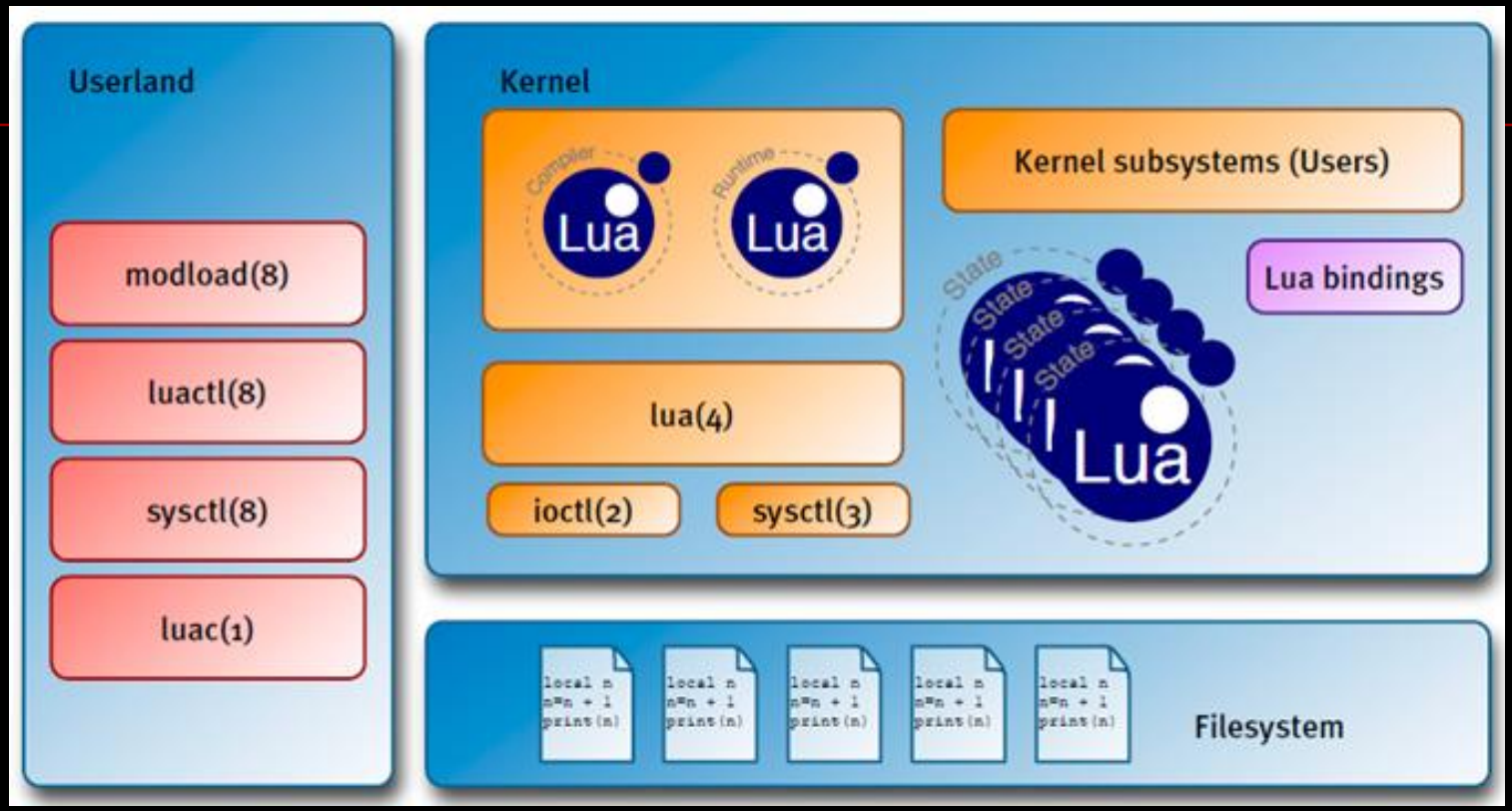
- http://www.brendangregg.com/blog/index.html



Source: https://github.com/iovisor/bcc/

Kernel Development

- NetBSD Kernel scripting with Lua



Source: https://archive.fosdem.org/2013/schedule/event/lua_in_the_netbsd_kernel/

- Deliver a higher-level programming environment to the Kernel
- Great innovation in OS development

8) Pros & Cons

Pros

- Could replace lots of debugfs files
- No need kernel debug symbols
- Scalable for dynamic probing
- Lower performance impact than even perf events
- Security: sandboxing + verifier
- …

Cons

- Up to 512 bytes stack
- Max 4096 instructions per program
- No more than 64 maps
- Call kernel functions with up to 5 arguments
- There is only one eBPF program (== one eBPF main routine) and it cannot call other eBPF functions
- …

II. XDP (eXpress Data Path)

1) Overview

- The https://www.iovisor.org/technology/xdp
- https://lwn.net/Articles/708087/ //Debating the value of XDP
- Generic hook
- eBPF-based "In-Kernel DPDK"

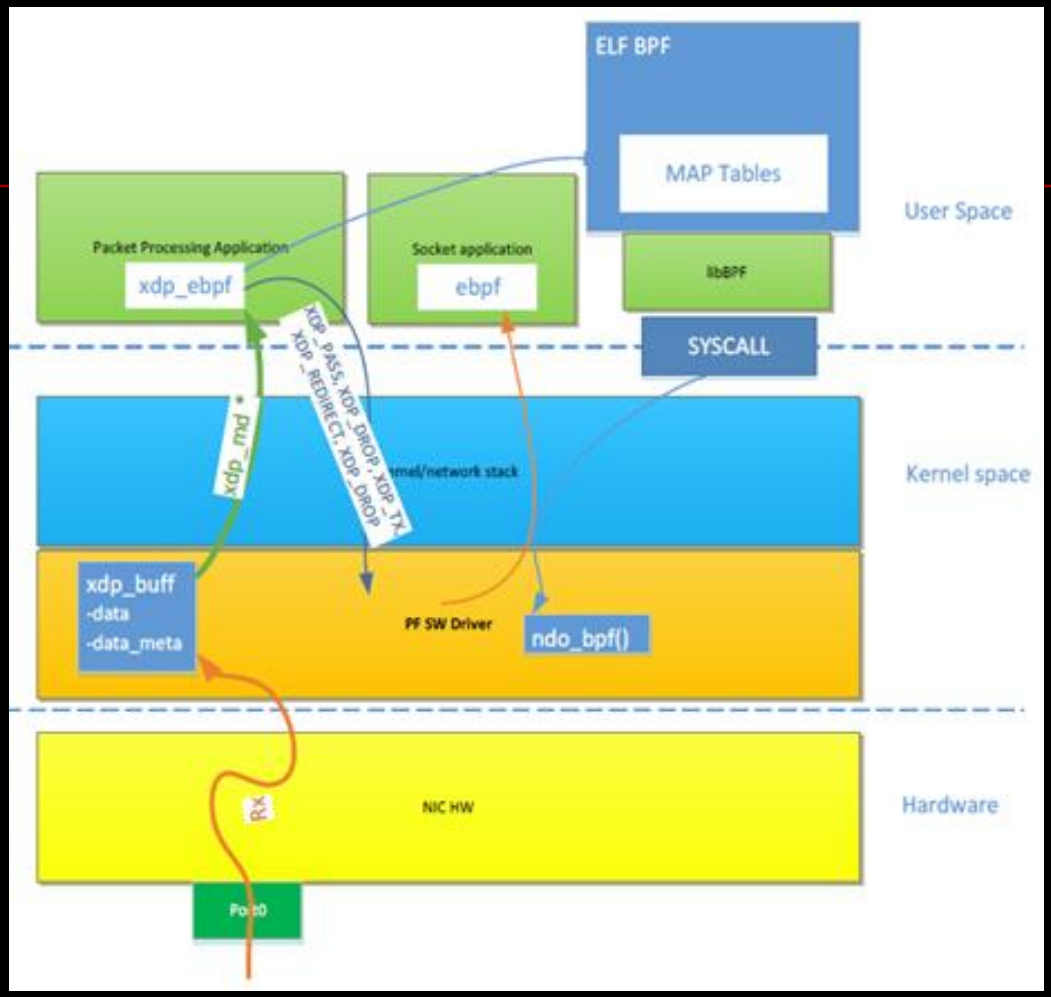
XDP is a further step in evolution and enables to run a specific flavor of BPF programs from the network driver with direct access to the packet's DMA buffer. This is, by definition, the earliest possible point in the software stack, where programs can be attached to in order to allow for a programmable, high performance packet processor in the Linux kernel networking data path.

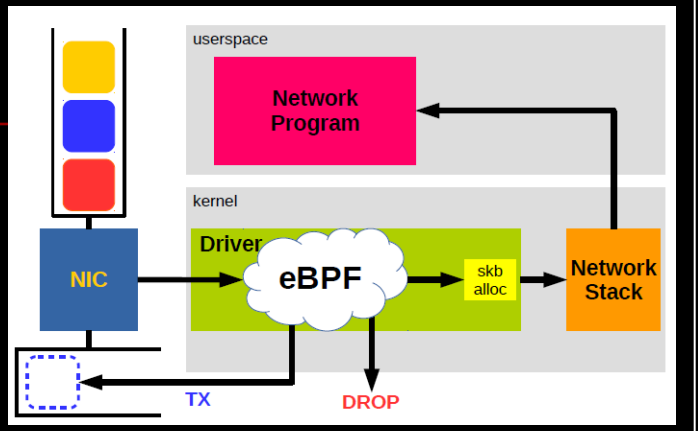
Source: https://github.com/cilium/cilium

- Works in concert with the kernel and its infrastructure (!)
- Advantages of XDP
  - Reuses upstream kernel drivers and tooling
  - Same security model as kernel for accessing hardware
  - Allows for flexible structuring of workloads
  - Punting to stable, efficient TCP/IP stack already available
  - No need for crossing boundaries when punting to sockets
  - No third party code/licensing required to use it
  - Shipped everywhere since kernel 4.8

Source: https://ossna2017.sched.com/event/BCsg/making-the-kernels-networking-data-path-programmable-with-bpf-and-xdp-daniel-borkmann-covalent

Software Model





Source: https://www.slideshare.net/lcplcp1/xdp-and-ebpfmaps

- eBPF trigger actions based on return codes
  - **XDP_DROP** - very fast drop by recycling
    - DDoS mitigation
  - **XDP_PASS** – pass possibly modified packet to network stack
    - Handle and pop new unknown encap protocols
  - **XDP_TX** – Transmit packet back out same interface
    - Facebook use it for load-balancing, and DDoS scrubber
  - **XDP_ABORTED** – also drop, but indicate error condition
    - Tracepoint: xdp_exception
  - **XDP_REDIRECT** – Transmit out other NICs
    - Very new (est.4.14), (plan also use for steering packets CPUs + sockets)

Source: http://people.netfilter.org/hawk/presentations/theCamp2017/theCamp2017_XDP_eBPF_technology_Jesper_Brouer.pdf

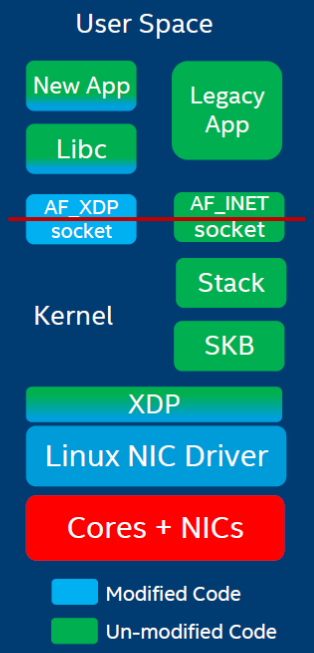
Source: Accelerating Load Balancing programs using HW-Based Hints in XDP (OCP 2018)

2) AF_XDP

- https://lwn.net/Articles/750293/
- https://lwn.net/Articles/750845/
- https://patchwork.ozlabs.org/cover/867937/

Proposed Solution

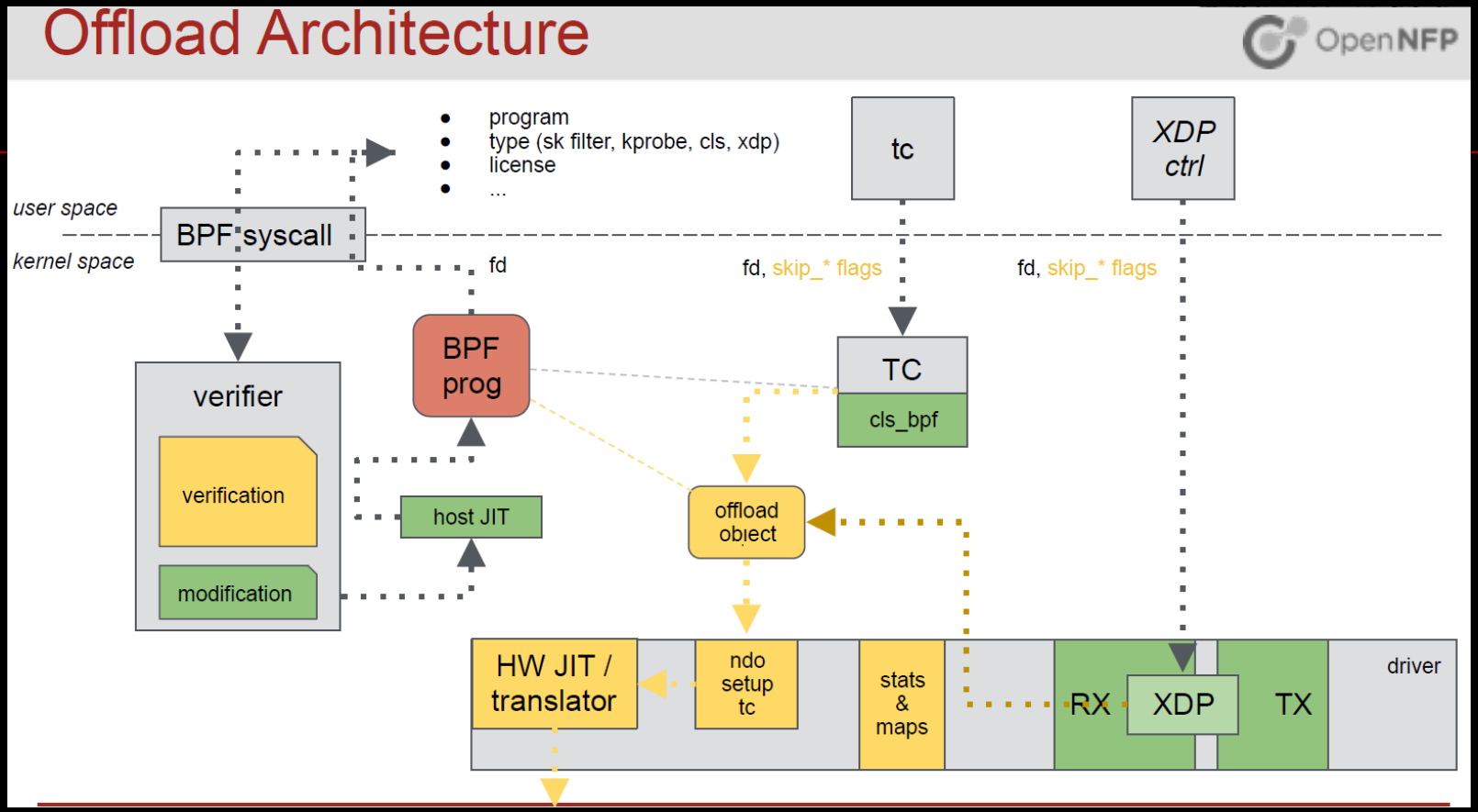
- New fast packet interfaces in Linux
  - AF_XDP: XDP's user-space interface
  - No system calls in data path
  - True zero-copy mode with new allocator, DMA packet buffers mapped to user space
  - Copy-mode for non-modified drivers
  - HW descriptors only mapped to kernel
- ZC mode requires HW steering support for untrusted applications
  - Copy required otherwise
- Goal is to hit 40 Gbit/s line rate on a single core for large packets and 25 Gbit/s for 64 byte packets



Intel Public Network Platforms Group intel

Source: Accelerating Load Balancing programs using HW-Based Hints in XDP (Fosdem 2018)

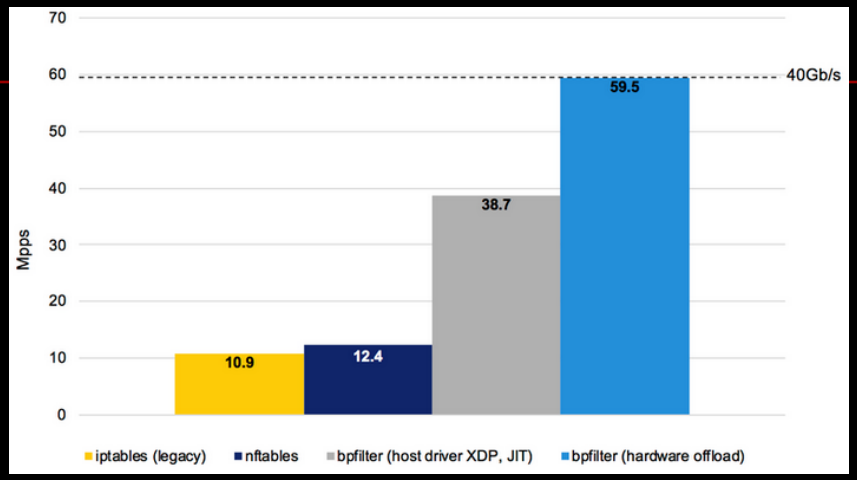
3) Offloading



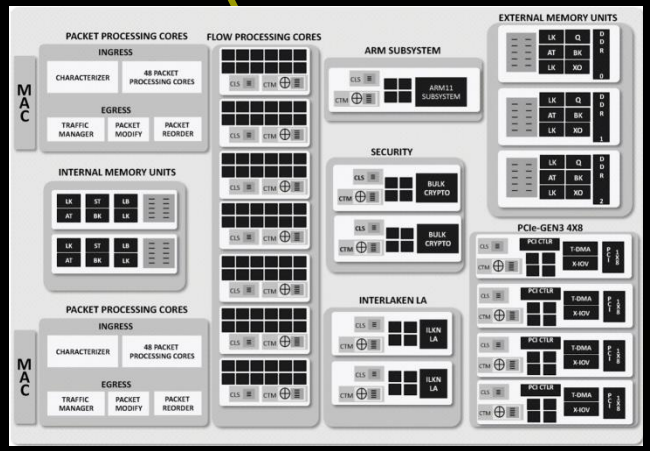
Source: https://www.slideshare.net/Open-NFP/transparent-ebpf-offload-playing-nice-with-the-linux-kernel

Performance

- **https://www.netronome.com/blog/frnog-30-faster-networking-la-francaise/**



- **The NFP (Network Flow Processor) Architecture**

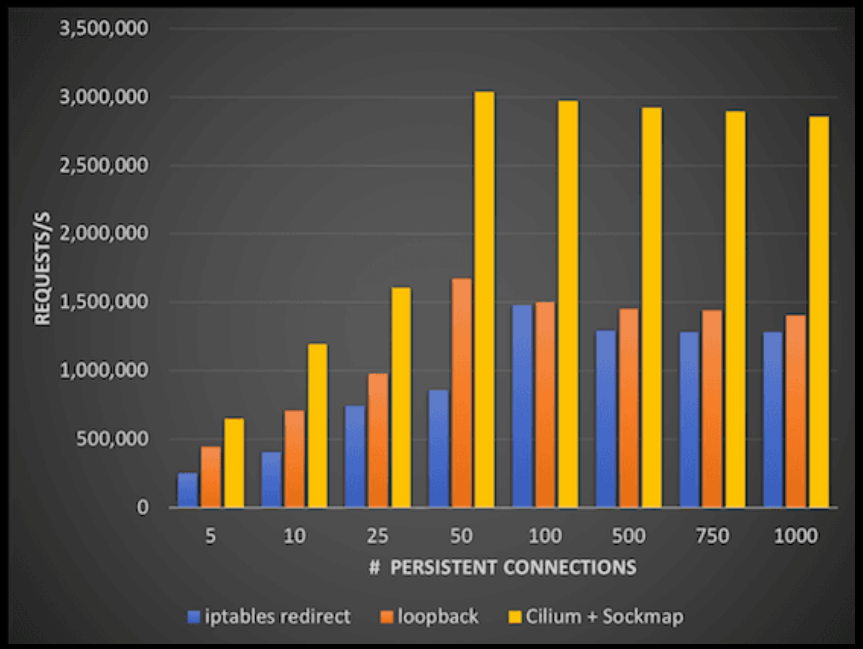


Source: https://www.netronome.com/media/documents/eBPF_HW_OFFLOAD_HNiMne8_2_.pdf

III. eBPF-driven Secure Network

Overview

- https://lwn.net/Articles/747551/ //BPF comes to firewalls
- ~~https://cilium.io/blog/2018/04/17/why-is-the-kernel-community-replacing-iptables/~~
- https://cilium.io/blog/2018/04/24/cilium-security-for-age-of-microservices/
- Flexibility, Scalability, Performance…



1) Cilium

Overview

- https://github.com/cilium/
- HTTP, gRPC, and Kafka Aware Security and Networking for Containers with BPF & XDP

Cilium is open source software for providing and transparently securing network connectivity and loadbalancing between application containers and services deployed using Linux container management platforms like Docker and Kubernetes.

A new Linux kernel technology called eBPF is at the foundation of Cilium, which enables the dynamic insertion of BPF bytecode into the Linux kernel. Cilium generates eBPF programs for each individual application container to provide networking, security, loadbalancing and visibility.

- http://docs.cilium.io/en/doc-1.0/kubernetes/
- Cilium 1.1 (https://cilium.io/blog/2018/06/26/cilium-11/):

  Deep Istio Integration
  Support for additional container runtimes
  Additional Network Security for Kubernetes
  Extended IP/CIDR policy enforcement capabilities
  Improved connection tracker efficiency
  Efficiency & Scale
  Additional Prometheus metrics
  Reliability Work
  Operations
  Documentation

eBPF Code Generation at Container Startup

- **Generate networking code at container startup, and tailored to each individual container**

On the fly BPF program generation means:

- Extensibility of userspace networking in the kernel
- MAC, IP, port number, ... all become constants
  → compiler can optimize heavily!
- BPF programs can be recompiled and replaced without interrupting the container and its connections
  - Features can be compiled in/out at runtime with container granularity
- Access to fast BPF maps and perf ring buffer to interact with userspace.
  - Drop monitor in *n*Mpps* context
  - Use notifications for policy learning, IDS, logging, ...

Source: "Cilium: Fast IPv6 Container Networking with BPF and XDP" LinuxCon 2016, Toronto

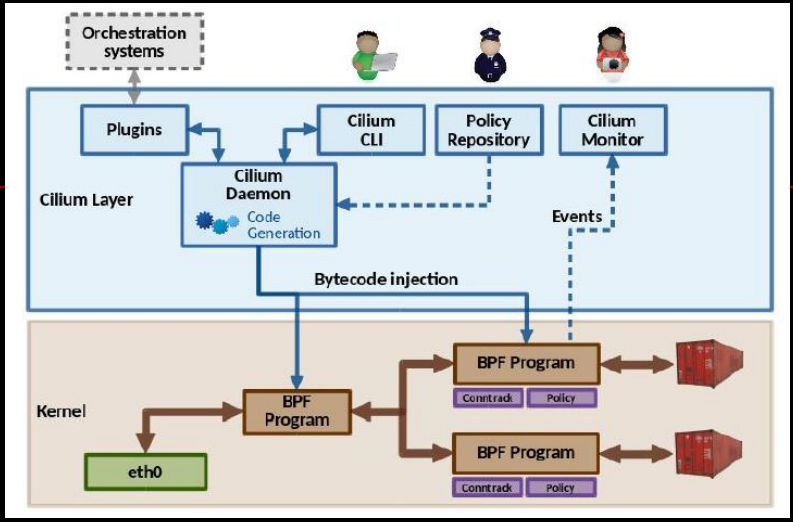
- **https://godoc.org/github.com/cilium/cilium/pkg/bpf**
- **$CILIUM_SRC/bpf**

```
[myrpi4@promote cilium]$ tree -L 1 bpf
bpf
├── bpf_features.h
├── bpf_lb.c
├── bpf_lxc.c
├── bpf_netdev.c
├── bpf_overlay.c
├── bpf_xdp.c
├── cilium-map-migrate.c
├── COPYING
├── filter_config.h
├── include
├── init.sh
├── join_ep.sh
├── lib
├── lxc_config.h
├── Makefile
├── netdev_config.h
├── node_config.h
├── probes
├── run_probes.sh
└── spawn_netns.sh
```

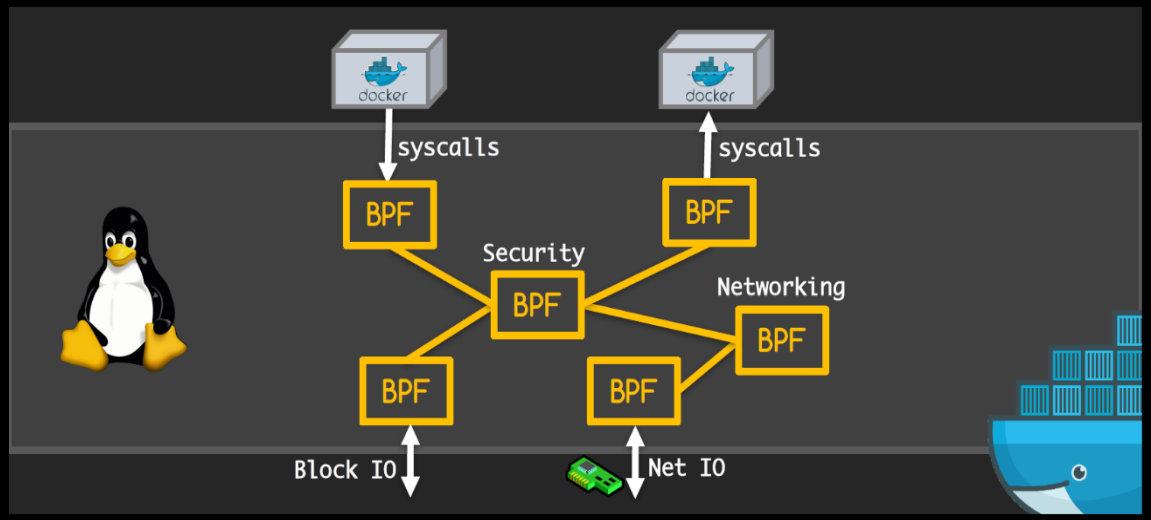
```
[myrpi4@promote cilium]$ tree -L 1 bpf/lib
bpf/lib
├── arp.h
├── common.h
├── conntrack.h
├── csum.h
├── dbg.h
├── drop.h
├── encap.h
├── eps.h
├── eth.h
├── events.h
├── icmp6.h
├── ipv4.h
├── ipv6.h
├── l3.h
├── l4.h
├── lb.h
├── lxc.h
├── maps.h
├── metrics.h
├── nat46.h
├── policy.h
├── trace.h
├── utils.h
└── xdp.h
```

```
[myrpi4@promote cilium]$ tree -L 1 bpf/probes/
bpf/probes/
├── raw_change_tail.t
├── raw_insn.h
├── raw_invalidate_hash.t
├── raw_lpm_map.t
├── raw_lru_map.t
├── raw_main.c
├── raw_map_val_adj.t
└── raw_mark_map_val.t
```

- ***Arch***



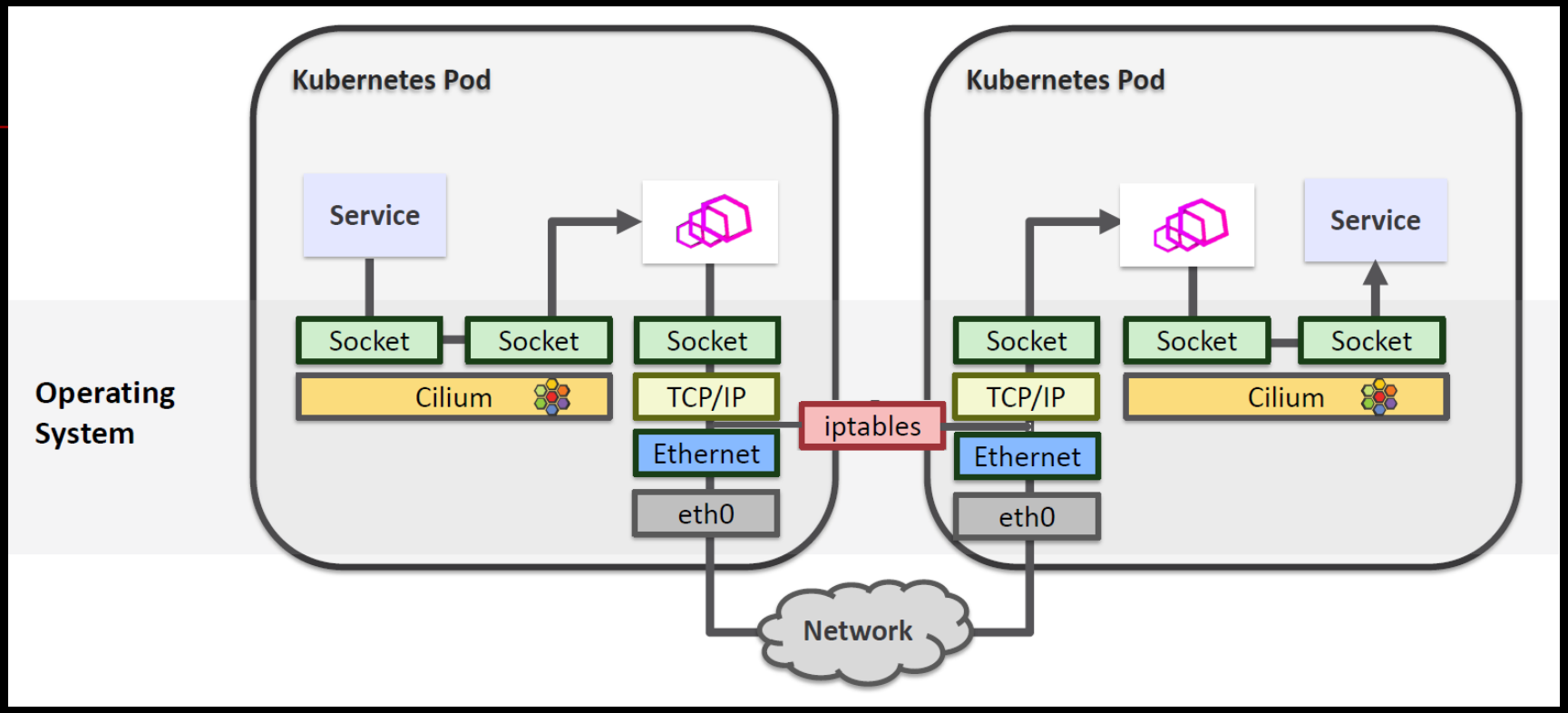
Source: https://www.slideshare.net/ThomasGraf5/cilium-fast-ipv6-container-networking-with-bpf-and-xdp



Source: https://www.slideshare.net/ThomasGraf5/dockercon-2017-cilium-network-and-application-security-with-bpf-and-xdp

Sidecar Injection

- https://istio.io/docs/reference/commands/sidecar-injector/



Source: Accelerating Envoy and Istio with Cilium and the Linux Kernel (KubeCon EU 2018)

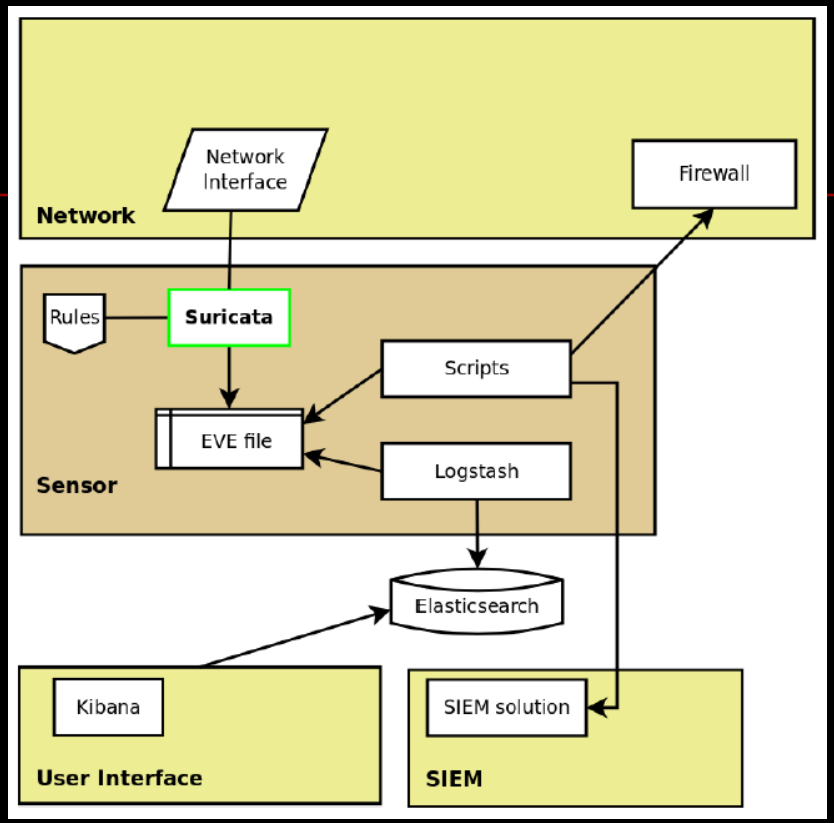
2) Suricata

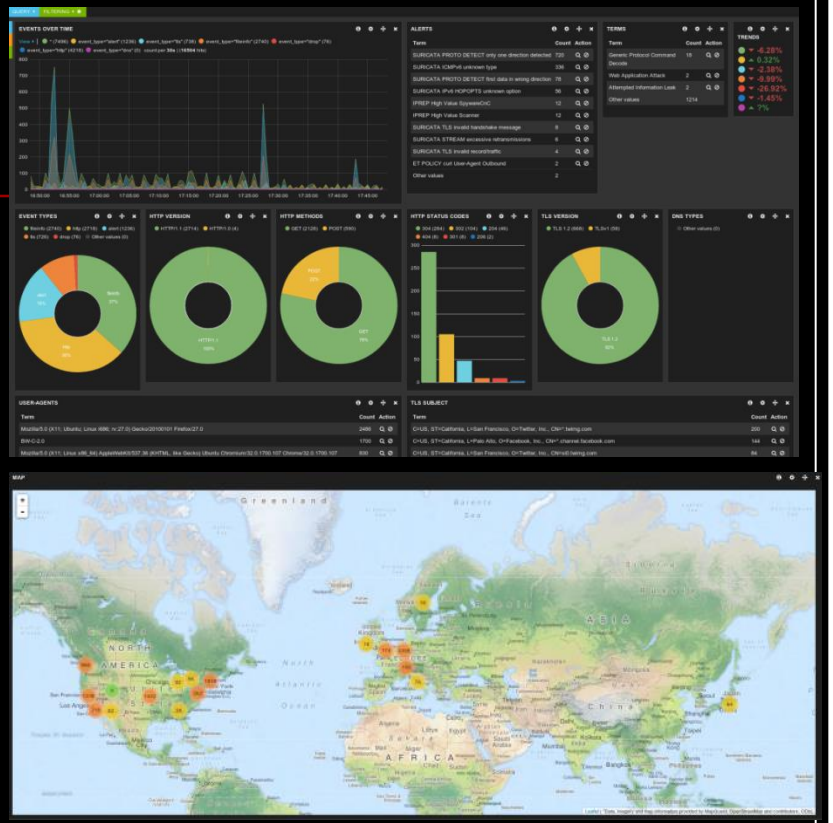
- https://suricata-ids.org/
- a network intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM)

- https://lwn.net/Articles/737771 Using eBPF and XDP in Suricata
- http://suricata.readthedocs.io/en/latest/capture-hardware/ebpf-xdp.html?highlight=XDP
- You can handle the packets…

Source: Suricata Extreme Performance Tuning (SuriCon 2017)

Ecosystem

- 



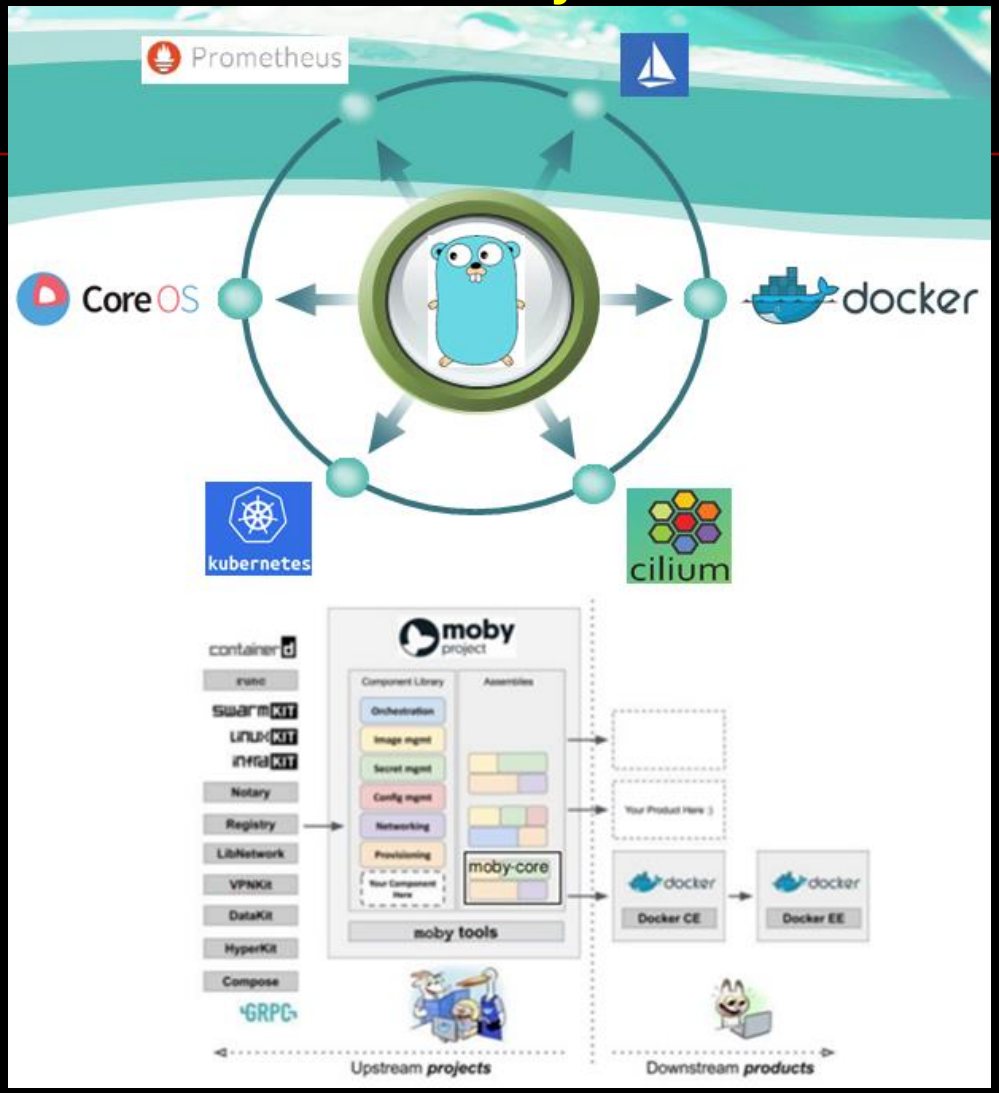
Source: eBPF and XDP seen from the eyes of a meerkat (Kernel Recipes 2017)

Future

- https://suricata-ids.org/2017/08/01/rust-and-suricata/
- https://github.com/qmonnet/rbpf

3) Interaction with Go

- Go-based Cloud Ecosystem



gobpf

- https://github.com/iovisor/gobpf
- Go bindings for the BCC framework as well as low-level routines to load and use eBPF programs from .elf files
- https://kinvolk.io/blog/2017/09/an-update-on-gobpf---elf-loading-uprobes-more-program-types

eBPF

- https://github.com/newtools/ebpf
- go library that provides utilities for loading, compiling, and debugging eBPF programs

cgnet

- https://github.com/kinvolk/cgnet
- uses eBPF to gather network statistics from cgroups

Using eBPF to analysis Go program

- http://www.brendangregg.com/blog/2017-01-31/golang-bcc-bpf-function-tracing.html

4) bpfilter

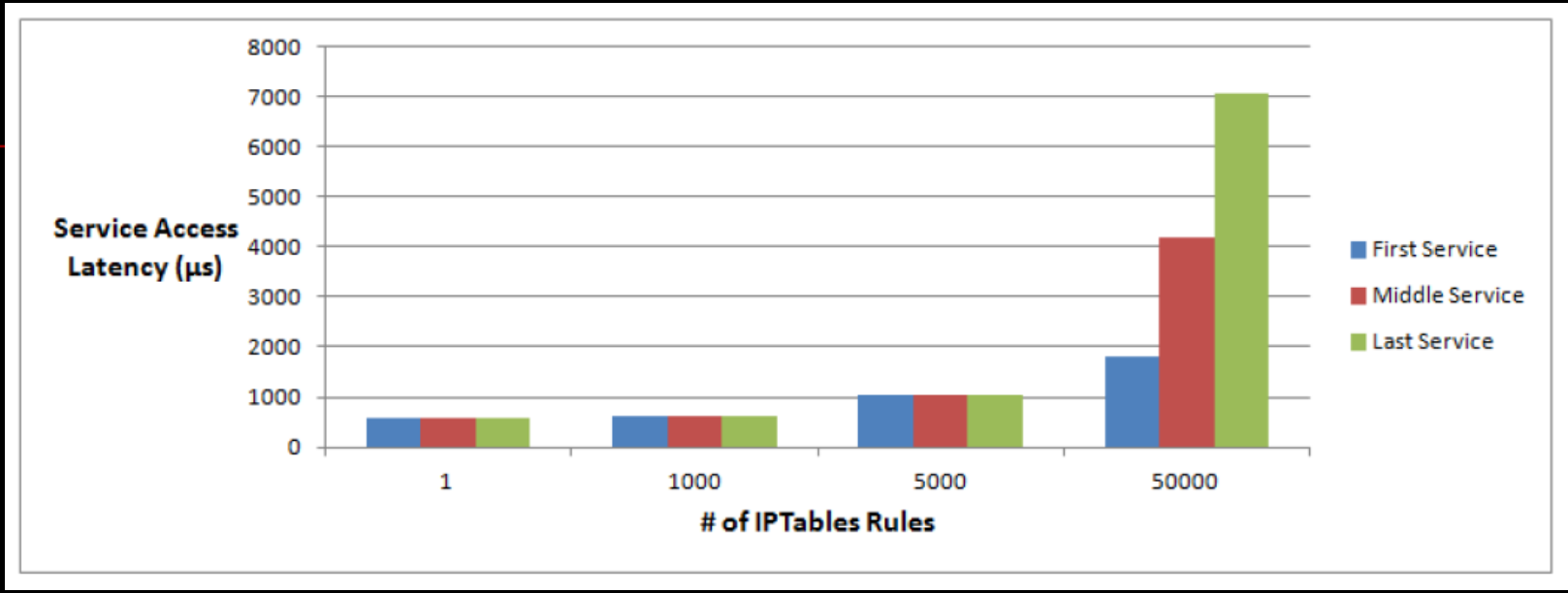
- https://www.mail-archive.com/netdev@vger.kernel.org/msg217095.html
- https://lwn.net/Articles/747504/ //net: add bpfilter
- https://lwn.net/Articles/749113/ //Re: [PATCH net-next] modules: allow modprobe load regular elf binaries
- https://lwn.net/Articles/749108/ //Designing ELF modules
- https://www.spinics.net/lists/netdev/msg486873.html //[RFC,POC 1/3] bpfilter: add experimental IMR bpf translator
- https://www.mail-archive.com/netdev@vger.kernel.org/msg217425.html [PATCH RFC PoC 0/3] nftables meets bpf
- ...
- time to replace iptables/nftables/netfiler with eBPF/bpfilter !

  https://lwn.net/Articles/755919/ //bpfilter (and user-mode blobs) for 4.18

  https://git.kernel.org/pub/scm/linux/kernel/git/davem/net-next.git/log/?qt=grep&q=bpfilter

Limitations of iptables

- Latency (lack of incremental updates)



Source: Scale Kubernetes to Support 50,000 Services (LC3 Beijing 2017)

- Not a good fit for Containers

IV. eBPF on ARM

1) ARM Development Boards

Raspberry Pi

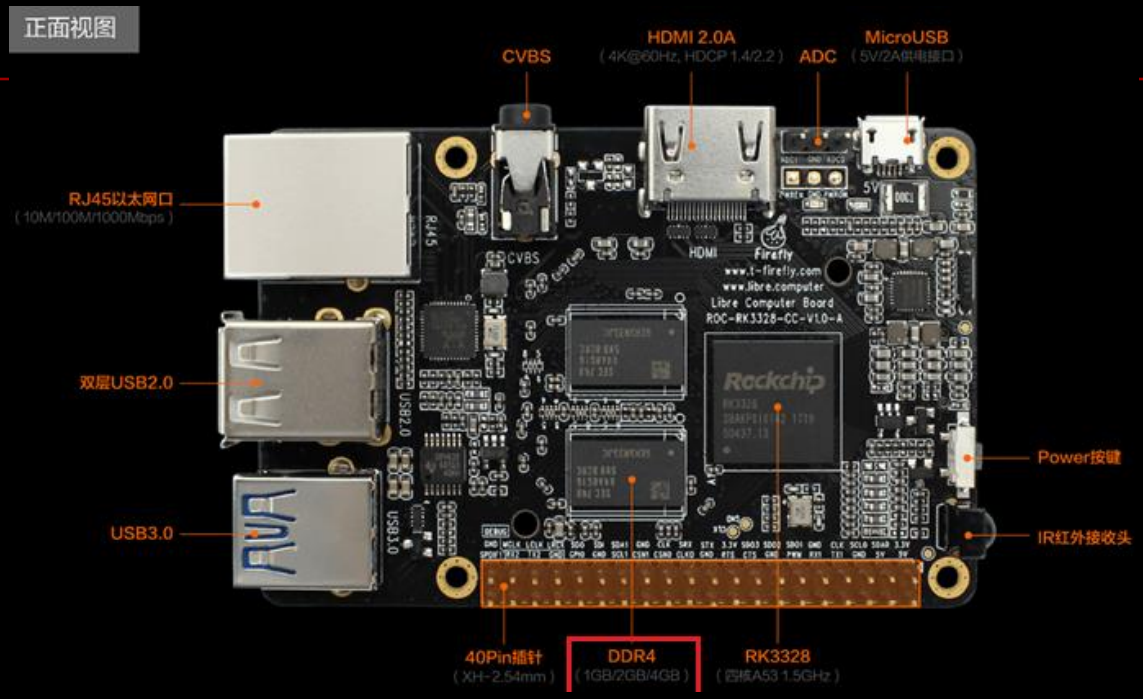
- https://www.raspberrypi.org/
- https://en.wikipedia.org/wiki/Raspberry_Pi

	Model 3 B	Model 3 B+
Release date	Feb, 2016	Mar, 2018
Arch	ARMv8-A	ARMv8-A
SoC	BCM2837	BCM2837B0
CPU	1.2 GHz 64-bit quad-core ARM Cortex-A53	1.4 GHz 64-bit quad-core ARM Cortex-A53
GPU	VideoCore IV	VideoCore IV
Memory (SDRAM)	1GB LPDDR2 RAM @900MHz (shared with GPU)	1GB LPDDR2 RAM @900MHz (shared with GPU)
Network	10/100 Mbit/s Ethernet, 802.11n wireless, Bluetooth 4.1	10/100/1000 Mbit/s Ethernet (real speed ~300 Mbit/s), 802.11ac dual band 2.4/5 GHz wireless, Bluetooth 4.2 LS BLE

- Official release (Raspbian with Linux Kernel 4.14 currently) still does not support AArch64

ROC-RK3328-CC

- http://www.t-firefly.com/product/rocrk3328cc.html
- http://opensource.rock-chips.com/wiki_RK3328



- my testing board has **4GB DDR4 @2133MHz**
- Ubuntu 16.04/Debian 9/Android 7.1.1 for AARCH64

2) Distributions for AARCH64

Debian (HypriotOS-RPi64)

- https://blog.hypriot.com/post/building-a-64bit-docker-os-for-rpi3
  https://github.com/dieterreuter/workshop-raspberrypi-64bit-os

```
HypriotOS/arm64: pirate@black-pearl in ~
$ uname -a
Linux black-pearl 4.9.13-bee42-v8 #1 SMP PREEMPT Fri Mar 3 16:42:37 UTC 2017 aarch64 GNU/Linux
```

- lightweight, and optimized for Docker, but is prone to have unmet dependencies

```
The following packages have unmet dependencies:
 libmariadb-dev-compat : Conflicts: libmariadbclient-dev but 10.1.26-0+deb9u1 is to be installed
                         Conflicts: libmariadbclient-dev-compat but 10.1.26-0+deb9u1 is to be installed
 libmariadbclient-dev : Depends: libmariadbclient18 (= 10.1.26-0+deb9u1) but 1:10.1.29-6+b1 is to be installed
 libmariadbclient-dev-compat : Conflicts: libmariadb-client-lgpl-dev-compat
                               Conflicts: libmariadb-dev-compat but 2.3.2-2 is to be installed
E: Unable to correct problems, you have held broken packages.
```

Fedora

- out-of-box support for RPi64 since Fedora 27
- 

- "heavyweight", but with upstream kernel, toolchain, and packages support, a better package management system
- http://fedoraproject.org/wiki/Objectives/Fedora_IoT

...

3) My Practice

Kernel

- https://github.com/iovisor/bcc/blob/master/INSTALL.md

Required

```
CONFIG_BPF=y
CONFIG_BPF_SYSCALL=y
# [optional, for tc filters]
CONFIG_NET_CLS_BPF=m
# [optional, for tc actions]
CONFIG_NET_ACT_BPF=m
CONFIG_BPF_JIT=y
CONFIG_HAVE_BPF_JIT=y
# [optional, for kprobes]
CONFIG_BPF_EVENTS=y
```

Optional

```
CONFIG_NET_SCH_SFQ=m
CONFIG_NET_ACT_POLICE=m
CONFIG_NET_ACT_GACT=m
CONFIG_DUMMY=m
CONFIG_VXLAN=m
```

- **HypriotOS-RPi64**

```
CONFIG_BPF=y
CONFIG_BPF_SYSCALL=y
#BPF_JIT_ALWAYS_ON is not set
CONFIG_BPF_JIT=y
CONFIG_HAVE_BPF_JIT=y
CONFIG_HAVE_EBPF_JIT=y
CONFIG_CGROUP_BPF=y
# [optional, for kprobes]
CONFIG_BPF_EVENTS=y
# [optional, for tc filters]
CONFIG_NET_CLS_BPF=m
# [optional, for tc actions]
CONFIG_NET_ACT_BPF=m
# [optional, for Xtables matches]
CONFIG_NETFILTER_XT_MATCH_BPF=m
# [optional, for using a stream parser with BPF_MAP_TYPE_SOCKMAP]
CONFIG_BPF_STREAM_PARSER=y
# [optional, for execute BPF program as route nexthop action]
CONFIG_LWTUNNEL_BPF=y
# [optional, for testing]
CONFIG_TEST_BPF=m
```

Fedora Minimal 28 AARCH64

```
[myrpi4@promote boot]$ cat config-4.16.15-300.fc28.aarch64 |grep BPF
CONFIG_CGROUP_BPF=y
CONFIG_BPF=y
CONFIG_BPF_SYSCALL=y
CONFIG_BPF_JIT_ALWAYS_ON=y
CONFIG_NETFILTER_XT_MATCH_BPF=m
CONFIG_NET_CLS_BPF=m
CONFIG_NET_ACT_BPF=m
CONFIG_BPF_JIT=y
CONFIG_BPF_STREAM_PARSER=y
CONFIG_LWTUNNEL_BPF=y
CONFIG_HAVE_EBPF_JIT=y
CONFIG_BPF_EVENTS=y
# CONFIG_TEST_BPF is not set
```

- https://github.com/raspberrypi/linux (branch rpi-4.17.y)
- on RPi 3B with HypriotOS-RPi64 v20180429 + GCC 7.3.0-21 + gnu ld 2.30 + jemalloc 5.1.0 + 5GB Memory (1GB DDR3 + 4GB Swap) and JOBS=4:

  ~3h16m for a full build

```
HypriotOS/arm64: pirate@black-pearl in /
$ uname -a
Linux black-pearl 4.17.0-v8+ #1 SMP PREEMPT Wed Jun 13 14:24:07 UTC 2018 aarch64 GNU/Linux
```

LLVM

- http://llvm.org/docs/GettingStarted.html
- http://llvm.org/docs/CMake.html
- http://llvm.org/docs/HowToBuildOnARM.html
- https://bugs.llvm.org/show_bug.cgi?id=37668 //_Float16
- https://bugs.llvm.org/show_bug.cgi?id=37725 //Attributes.inc
- on ROC-RK3328-CC with Debian 9 + Kernel 4.4.114 + GCC 7.3.0-19 + gnu gold 1.15 + jemalloc 5.1.0 + 6GB Memory (4GB DDR4 + 2GB Swap) and ninja 1.8.2 JOBS=4:

  ~7h20m for a full build

```
/opt/MyWorkSpace/DevSW/Toolchain/LLVM/7.0.0/bin/clang-7: /lib64/libtinfo.so.6: no version information available (required
 by /opt/MyWorkSpace/DevSW/Toolchain/LLVM/7.0.0/bin/clang-7)
```

BCC

- build BCC with **GCC**
  on **RPi 3B** with HypriotOS-RPi64 v20180429 + GCC 7.3.0-23 + gnu ld 2.30 + jemalloc 5.1.0 + 5GB Memory (1GB DDR3 + 4GB Swap) and **JOBS=1:**

```
HypriotOS/arm64: pirate@black-pearl in /opt/MyWorkSpace/DevSW/Toolchain
$ free -m
              total        used        free      shared  buff/cache   available
Mem:            968         941           4           0          22          11
Swap:          4095         351        3744
```

  ~1h for a full build on BCC v0.6.0

- build BCC with latest **LLVM 7.0.0**
  on **RPi 3B+** with Fedora Minimal 28 + GCC 1.1.1-1 + lld 7.0.0 + jemalloc 5.1.0 + 7GB Memory (1GB DDR3 + 6GB Swap) and **JOBS=1:**
  //disable Lua: src/CMakeLists.txt

```
if(NOT PYTHON_ONLY)
add_subdirectory(cc)
endif()
if(ENABLE_CLANG_JIT)
add_subdirectory(python)
#add_subdirectory(lua)
endif()
```

//LLVM 7 built on ROC-RK3328-CC

/opt/MyWorkSpace/DevSW/Toolchain/LLVM/7.0.0

//Toolchain_Clang-LLVM-7-ARM64_For_eBPF-on-RPi3.cmake

```
set(CMAKE_SYSTEM_NAME Linux)
set(CMAKE_SYSTEM_PROCESSOR aarch64)

#set(tools /usr/lib/llvm-7)
set(tools /opt/MyWorkSpace/DevSW/Toolchain/LLVM/7.0.0)
set(CMAKE_C_COMPILER ${tools}/bin/clang)
set(CMAKE_CXX_COMPILER ${tools}/bin/clang++)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -I${tools}/include/c++/v1")
```

//cmake

```
cmake .. -DCMAKE_INSTALL_PREFIX=../../bcc-0.6.0_clang_out -DCMAKE_TOOLCHAIN_FILE=../../Toolchain_Clang-LLVM-7-ARM64_For_e
BPF-on-RPi3.cmake
```

//lld

```
[myrpi4@promote bin]$ sudo ln -sf /opt/MyWorkSpace/DevSW/Toolchain/LLVM/7.0.0/bin/ld.lld /usr/bin/ld
```

//patch for BCC v0.6.0: static_libstdc++.cmake

```
# only turn on static-libstdc++ if also linking statically against clang
string(REGEX MATCH ".*[.]a$" LIBCLANG_ISSTATIC "${libclangBasic}")
# if gcc 4.9 or higher is used, static libstdc++ is a good option
if (CMAKE_COMPILER_IS_GNUCC AND LIBCLANG_ISSTATIC)
  execute_process(COMMAND ${CMAKE_C_COMPILER} -dumpversion OUTPUT_VARIABLE GCC_VERSION)
  if (GCC_VERSION VERSION_GREATER 4.9 OR GCC_VERSION VERSION_EQUAL 4.9)
    execute_process(COMMAND ${CMAKE_C_COMPILER} -print-libgcc-file-name OUTPUT_VARIABLE GCC_LIB)
    get_filename_component(GCC_DIR "${GCC_LIB}" DIRECTORY)
    find_library(GCC_LIBSTDCPP libstdc++.a PATHS "${GCC_DIR}" NO_DEFAULT_PATH)
    if (GCC_LIBSTDCPP)
      message(STATUS "Using static-libstdc++")
      set(CMAKE_SHARED_LINKER_FLAGS "${CMAKE_SHARED_LINKER_FLAGS} -static-libstdc++")
    endif()
  endif()
else()
   message(STATUS "Using libc++ & libstdc++.a")
   set(CMAKE_SHARED_LINKER_FLAGS "${CMAKE_SHARED_LINKER_FLAGS} -stdlib=libc++ -L/usr/lib64,/usr/lib/gcc/aarch64-redhat-li
nux/8 -lc++abi -static-libstdc++")
endif()
```

//unfortunately, it still got failed:

```
/usr/bin/ld: error: undefined symbol: std::__1::cout
>>> referenced by FollyRequestContextSwitch.cc
>>>               CMakeFiles/FollyRequestContextSwitch.dir/FollyRequestContextSwitch.cc.o:(handle_output(void*, void*, in
t))

/usr/bin/ld: error: undefined symbol: std::__1::basic_ostream<char, std::__1::char_traits<char> >::operator<<(int)
>>> referenced by FollyRequestContextSwitch.cc
>>>               CMakeFiles/FollyRequestContextSwitch.dir/FollyRequestContextSwitch.cc.o:(handle_output(void*, void*, in
t))
```

…

Future

- **A fully customized Linux distribution:**
  - **Upstreaming Kernel**
  - **Full support for AARCH64**
  - **LLVM/Clang as the unique toolchain**
  - **Replace Glibc with Musl + JEMalloc**
  - **Meson based build system**
  - **Development related packages preinstalled**

  …

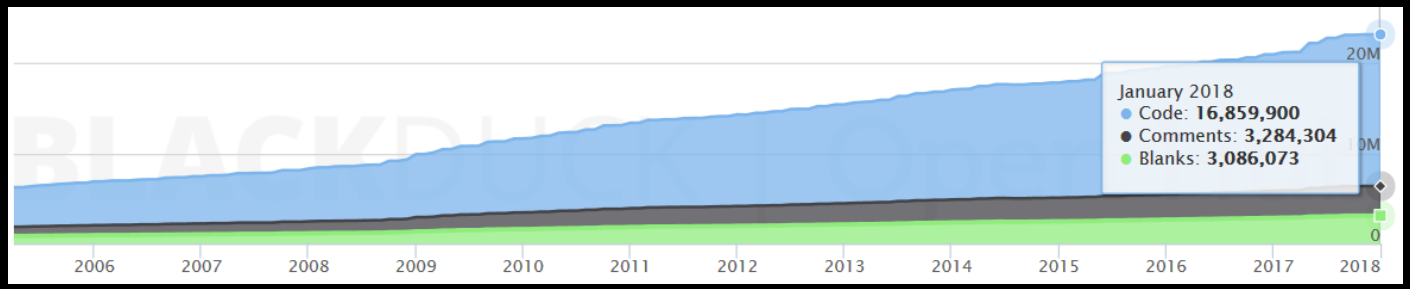
time to get rid of GNU?

V. Wrap-up

- Kernel Space & User Space Instrumentation

- Polyglot VM

  Changing the way you think about Linux Kernel development:



Source: https://www.openhub.net/p/linux/analyses/latest/languages_summary

- So:

  User Space/Kernel Space Repartition & Unifying

  Reconstruct nearly every aspect of Linux Networking and Security subsystem

Q & A

Thanks!

Reference

Slides/materials from many and varied sources:

- http://en.wikipedia.org/wiki/
- http://www.slideshare.net/
- https://www.kernel.org/doc/Documentation/
- http://man7.org/linux/man-pages/man2/bpf.2.html
- http://www.brendangregg.com/ebpf.html
- https://www.python.org
- http://llvm.org
- https://en.wikipedia.org/wiki/Just-in-time_compilation
- http://dpdk.org/
- https://www.netbsd.org/gallery/presentations/
- https://www.cncf.io/projects/
- https://www.opennetworking.org/
- https://www.opnfv.org/
- https://cilium.io/blog/2018/04/17/why-is-the-kernel-community-replacing-iptables/
- …