

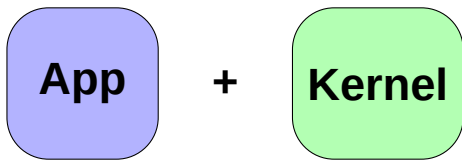


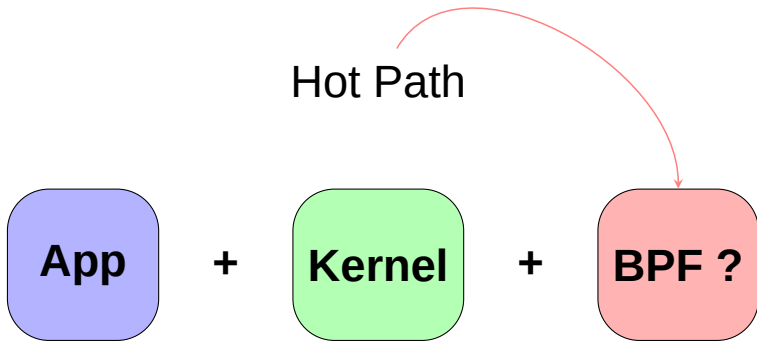
Performance Insights into eBPF Step by Step

Dmitrii Dolgov
Senior Software Engineer

14-09-2022







How to measure?

Global kernel stats

```
$ sysctl -w kernel.bpf_stats_enabled=1  
$ bpftool prog
```

```
379: raw_tracepoint [ ... ]  
run_time_ns 35875602162 run_cnt 160512637
```

Good old printk

```
// somewhere inside your BPF prog  
bpf_trace_printk("Timestamp: %lld", ts);
```

```
$ cat /sys/kernel/debug/tracing/trace_pipe  
$ bpftool prog tracelog
```

Probes before/after

```
kfunc:do_syscall_64 /comm = "prog"/  
{ @entry[kstack] = count(); }
```

```
kretfunc:do_syscall_64 /comm = "prog"/  
{ @return[kstack] = count(); }
```

Probes before/after

```
struct kretprobe {  
    struct kprobe kp;  
    kretprobe_handler_t handler;  
    kretprobe_handler_t entry_handler;  
    int maxactive;  
    int nmissed;  
    size_t data_size;  
    struct freelist_head freelist;  
    struct kretprobe_holder *rph;  
};
```


Probes before/after

```
SEC("fentry/XXX")
int BPF_PROG(fentry_XXX)
{
    // ...
}
```

```
SEC("fexit/XXX")
int BPF_PROG(fentry_XXX)
{
    // ...
}
```

Probes before/after

```
if (!btf) {  
    bpf_log(log,  
        "FENTRY/FEXIT program can only be"  
        "attached to another program"  
        "annotated with BTF\n");  
    return -EINVAL;  
}
```

Probes before/after

```
nopl    0x0(%rax,%rax,1)
```

```
xchg    %ax,%ax
```

```
push    %rbp
```

```
mov     %rsp,%rbp
```

```
sub     $0x20,%rsp
```

```
callq   0xffffffffffffe0096c
```

```
xchg    %ax,%ax
```

```
push    %rbp
```

```
mov     %rsp,%rbp
```

```
sub     $0x20,%rsp
```

Probes before/after

```
$ bpftool prog profile \  
    id 5 \  
    duration 10 \  
    cycles instructions
```

Profiling

```
Percent | uops_retired.stall_cycles
:
: if (duration_ns < min_duration_ns)
0.00 : 9f:movabs $0xfffffc90000009e000,%rdi
0.00 : a9:mov 0x0(%rdi),%rsi
:
: e = bpf_ringbuf_reserve( ... )
21.74 : ad:movabs $0xfffff888103e70e00,%rdi
0.00 : b7:mov $0xa8,%esi
0.00 : bc:xor %edx,%edx
0.00 : be:callq 0xfffffffffc0f9fbb8
```

How to improve?

BPF Instruction Set

```
if (pid < ts)
```

<pre>; if (pid < ts)</pre>		<pre>; if (pid < ts)</pre>
<pre> cmp %rsi,%rdi</pre>		<pre> cmp %rdi,%rsi</pre>
<pre> jae 0x000000000000000068</pre>		<pre> jbe 0x000000000000000065</pre>

eBPF Instruction Sets

BPF Instruction Set

```
$ llc probe.bc  
    -mcpu=v2  
    -march=+alu32  
    -march=bpf  
    -filetype=obj  
    -o probe.o  
# otherwise -mllvm -mcpu= ...
```


BPF 2 BPF

```
#ifndef __inline
# define __inline \
    inline __attribute__((always_inline))
#endif

static __inline int test_bpf2bpf(void) {}

# 0xfffffffffc1513a68:
    nopl    0x0(%rax,%rax,1)
    xor     %eax,%eax
    push    %rbp
# [ ... ]
```

BPF 2 BPF

```
static int test_bpf2bpf(void) { }
```

```
# 0xfffffffffc15b810c:
```

```
    nopl    0x0(%rax,%rax,1)
```

```
    xor     %eax,%eax
```

```
# [ ... ]
```

```
    callq   0x00000000000002370
```

```
# 0xfffffffffc15ba47c:
```

```
    nopl    0x0(%rax,%rax,1)
```

```
    xchg    %ax,%ax
```

```
# [ ... ]
```

BPF 2 BPF

```
if (idx && subprog[idx].has_tail_call && depth ≥ 256) {  
    verbose(env,  
        "tail_calls are not allowed when call stack"  
        "of previous frames is %d bytes. Too large\n",  
        depth);  
    return -EACCES;  
}
```

How to improve in the future?

Map batch operations

BPF_MAP_LOOKUP_BATCH

BPF_MAP_LOOKUP_AND_DELETE_BATCH

BPF_MAP_UPDATE_BATCH

BPF_MAP_DELETE_BATCH

BPF program pack allocator

```
struct bpf_prog_pack {  
    struct list_head list;  
    void *ptr;  
    unsigned long bitmap[];  
};  
  
// [ ... ]  
  
ro_header = bpf_prog_pack_alloc(  
    size, bpf_fill_ill_insns);
```

Bloom filter map


```
bpf_map_create(  
    BPF_MAP_TYPE_BLOOM_FILTER, NULL,  
    0, sizeof(value), 100, NULL);
```

Task local storage

```
ptr = bpf_task_storage_get(  
    &start, t, 0,  
    BPF_LOCAL_STORAGE_GET_F_CREATE);
```


Questions?

 @erthalion

 dmitrii.dolgov at redhat dot com