Linux Plumbers Conference | Richmond, VA | Nov. 13-15, 2023

BPF struct_ops - current status and the last developments

Kui-Feng Lee <kuifeng@meta.com>

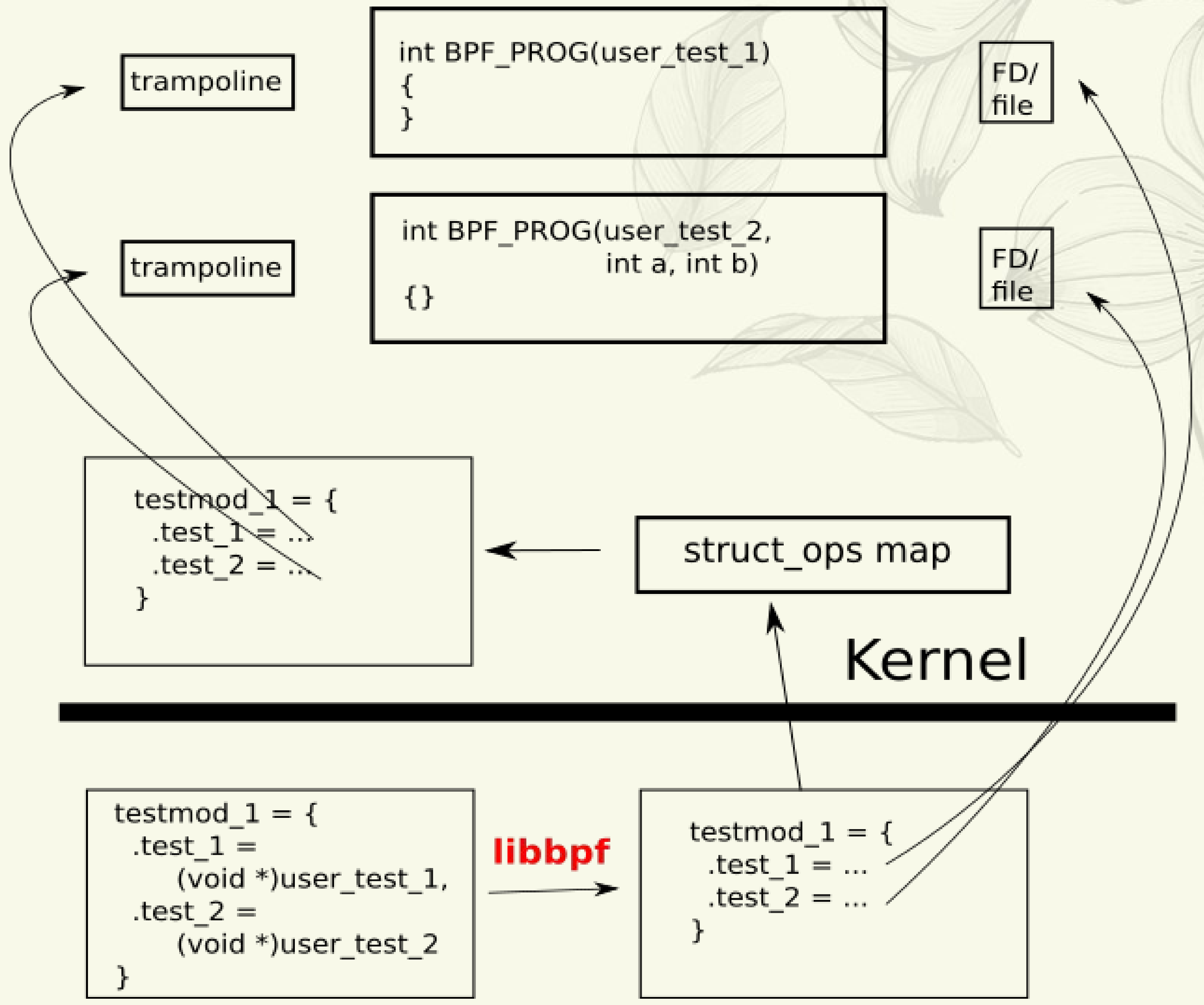
BPF struct_ops

The BPF struct_ops is a kernel-side feature in Linux which allows user-defined methods to be called by subsystems. For example, it is now possible to define a congestion control algorithm in BPF and then proceed to register it with the TCP subsystem in order to effectively regulate traffic.

```
struct bpf_testmod_ops {
        int (*test_1)(void);
        int (*test_2)(int a, int b);
};
```

```
SEC("struct_ops/user_test_1")
int BPF_PROG(user_test_1)
{
        return 0xdeadbeef;
}

SEC("struct_ops/user_test_2")
int BPF_PROG(user_test_2, int a, int b)
{
        return a + b;
}

SEC(".struct_ops.link")
struct bpf_testmod_ops testmod_1 = {
        .test_1 = (void *)user_test_1,
        .test_2 = (void *)user_test_2,
};
```

```
r = ops->test_2(4, 3)
```



```
int BPF_PROG(user_test_1)
{
}
```

```
int BPF_PROG(user_test_2,
             int a, int b)
{}
```

```
testmod_1 = {
  .test_1 = ...
  .test_2 = ...
}
```

```
testmod_1 = {
  .test_1 =
      (void *)user_test_1,
  .test_2 =
      (void *)user_test_2
}
```

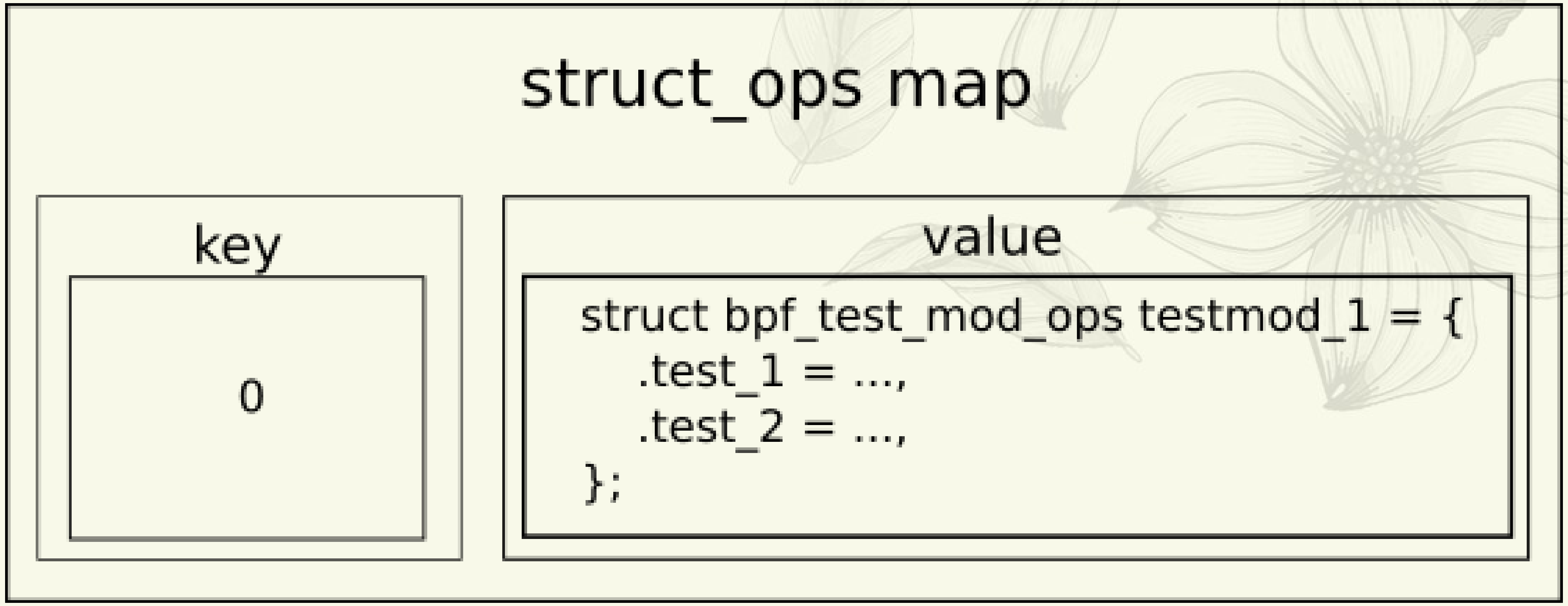
```
testmod_1 = {
  .test_1 = ...
  .test_2 = ...
}
```

bpf_dummy_struct_ops.c

TCP Congestion Control (tcp_ca) is the only consumer so far

New Developments

BPF links for struct_ops maps



```
struct bpf_test_mod_ops testmod_1 = {
    .test_1 = ...,
    .test_2 = ...,
};
```

The old API registers/unregisters a struct_ops map when its value is updated/deleted.

```
int ca1_cnt = 0;
int ca2_cnt = 0;

SEC("struct_ops/ca_update_1_init")
void BPF_PROG(ca_update_1_init, struct sock *sk)
{
        ca1_cnt++;
}

SEC("struct_ops/ca_update_cong_control")
void BPF_PROG(ca_update_cong_control, struct sock *sk,
            const struct rate_sample *rs)
{
}
 ……

SEC(".struct_ops.link")
struct tcp_congestion_ops ca_update_1 = {
        .init = (void *)ca_update_1_init,
        .cong_control = (void *)ca_update_cong_control,
        .ssthresh = (void *)ca_update_ssthresh,
        .undo_cwnd = (void *)ca_update_undo_cwnd,
        .name = "tcp_ca_update",
};
```

struct_ops types in kernel modules

```
#ifdef CONFIG_BPF_JIT
#ifdef CONFIG_NET
BPF_STRUCT_OPS_TYPE(bpf_dummy_ops)
#endif
#ifdef CONFIG_INET
#include <net/tcp.h>
BPF_STRUCT_OPS_TYPE(tcp_congestion_ops)
#endif
#endif
```

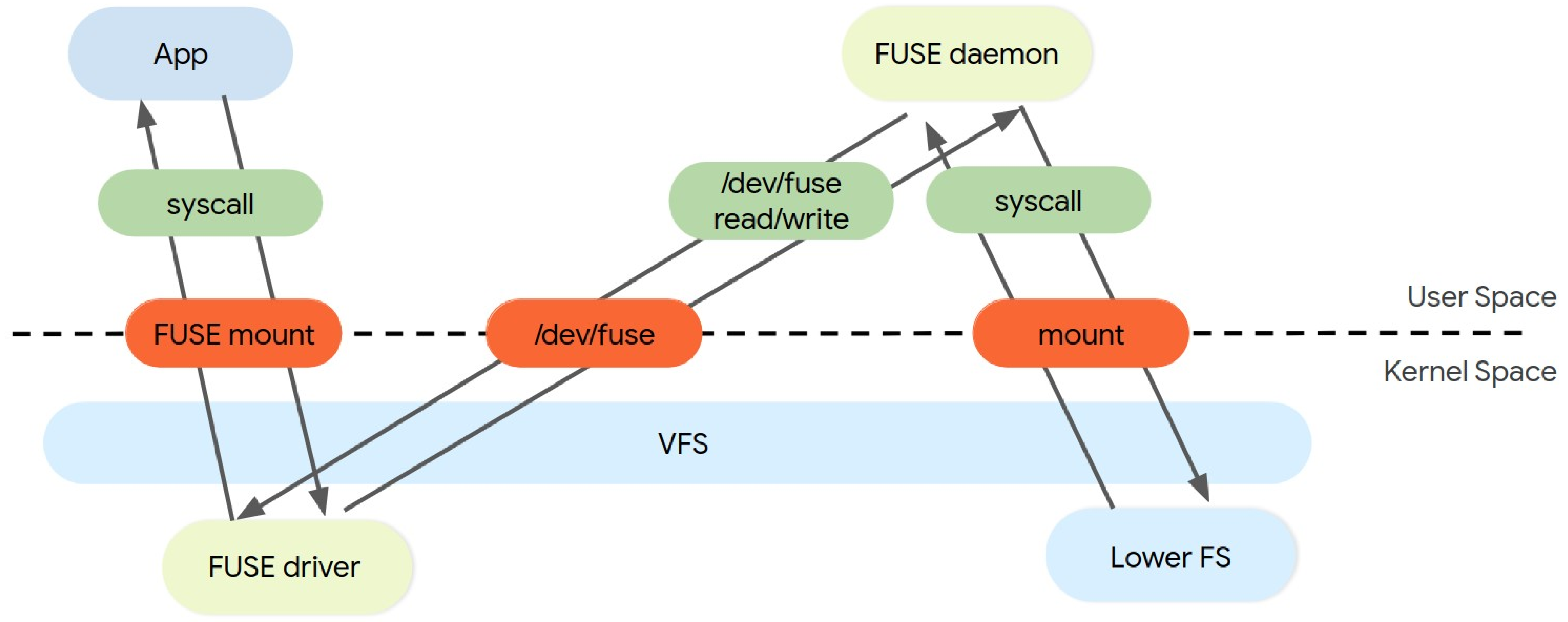
```
static struct bpf_struct_ops bpf_bpf_dummy_ops = {
	.verifier_ops = &bpf_dummy_verifier_ops,
	.init = bpf_dummy_init,
	.check_member = bpf_dummy_ops_check_member,
	.init_member = bpf_dummy_init_member,
	.reg = bpf_dummy_reg,
	.unreg = bpf_dummy_unreg,
	.name = "bpf_dummy_ops",
	.owner = THIS_MODULE,
};

static int __init bpf_dummy_struct_ops_init(void)
{
	return register_bpf_struct_ops(&bpf_bpf_dummy_ops);
}
```

Projects Going to Use struct_ops

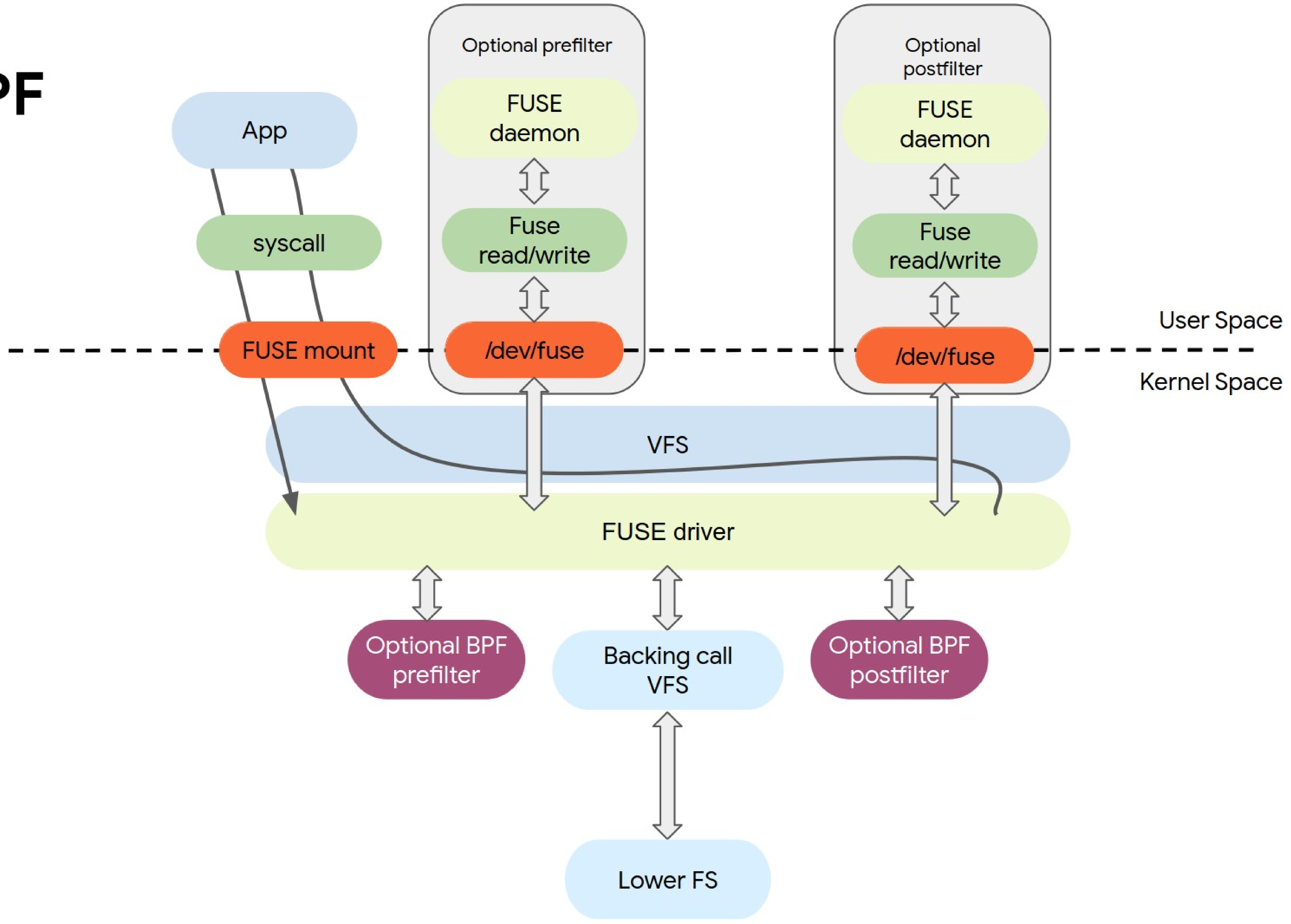
Fuse-BPF

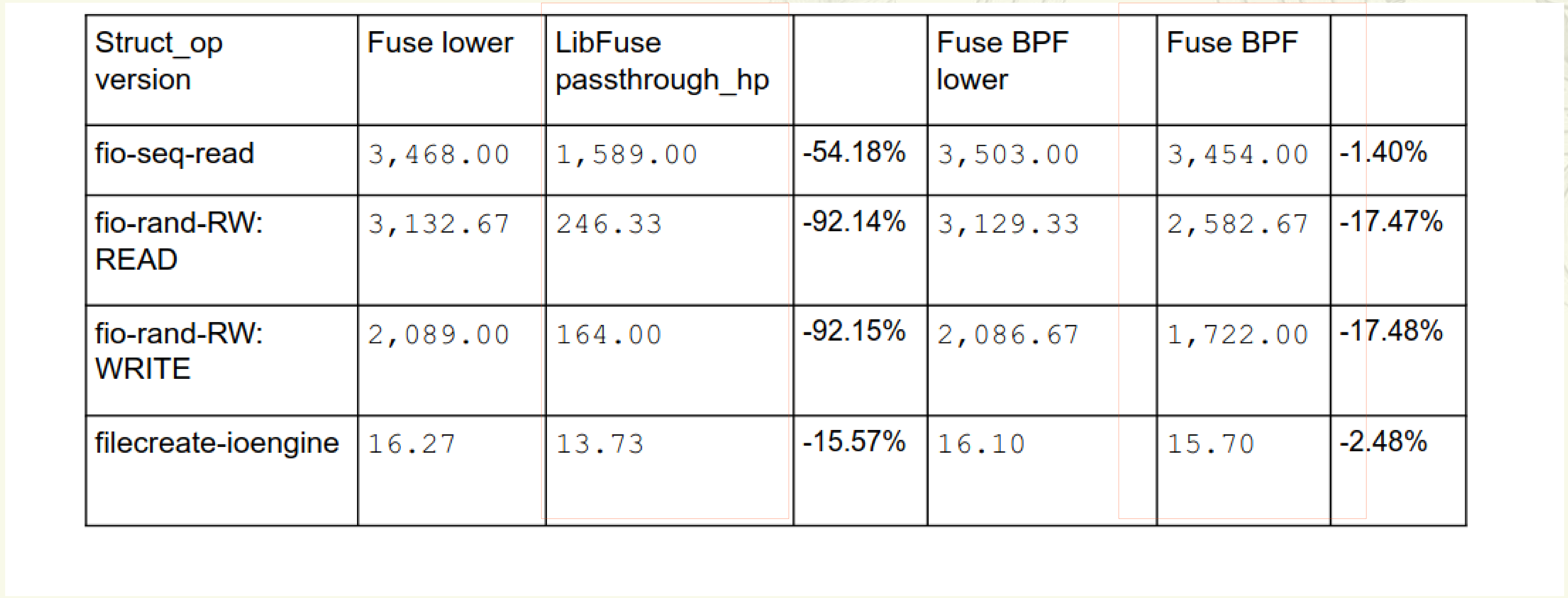
```
struct fuse_ops {
  uint32_t (*default_filter)(const struct bpf_fuse_meta_info *meta);
  uint32_t (*open_prefilter)(const struct bpf_fuse_meta_info *meta,
                             struct fuse_open_in *in);
  uint32_t (*open_postfilter)(const struct bpf_fuse_meta_info *meta,
                              const struct fuse_open_in *in,
                              struct fuse_open_out *out);
  uint32_t (*opendir_prefilter)(const struct bpf_fuse_meta_info *meta,
                                struct fuse_open_in *in);
  uint32_t (*opendir_postfilter)(const struct bpf_fuse_meta_info *meta,
                                 const struct fuse_open_in *in,
                                 struct fuse_open_out *out);
  ...
  char name[BPF_FUSE_NAME_MAX];
};
```

Classic Fuse



Fuse BPF


App
syscall
FUSE mount
Optional prefilter
FUSE daemon
Fuse read/write
/dev/fuse
Optional postfilter
FUSE daemon
Fuse read/write
/dev/fuse
User Space
Kernel Space
VFS
FUSE driver
Optional BPF prefilter
Backing call VFS
Optional BPF postfilter
Lower FS



Struct_op version	Fuse lower	LibFuse passthrough_hp		Fuse BPF lower	Fuse BPF	
fio-seq-read	3,468.00	1,589.00	-54.18%	3,503.00	3,454.00	-1.40%
fio-rand-RW: READ	3,132.67	246.33	-92.14%	3,129.33	2,582.67	-17.47%
fio-rand-RW: WRITE	2,089.00	164.00	-92.15%	2,086.67	1,722.00	-17.48%
filecreate-ioengine	16.27	13.73	-15.57%	16.10	15.70	-2.48%

sched_ext

```
s32 BPF_STRUCT_OPS(simple_init)
{
        if (!switch_partial)
                scx_bpf_switch_all();
        return 0;
}

void BPF_STRUCT_OPS(simple_enqueue, struct task_struct *p, u64 enq_flags)
{
        if (enq_flags & SCX_ENQ_LOCAL)
                scx_bpf_dispatch(p, SCX_DSQ_LOCAL, SCX_SLICE_DFL, enq_flags);
        else
                scx_bpf_dispatch(p, SCX_DSQ_GLOBAL, SCX_SLICE_DFL, enq_flags);
}

void BPF_STRUCT_OPS(simple_exit, struct scx_exit_info *ei)
{
        exit_type = ei->type;
}

SEC(".struct_ops")
struct sched_ext_ops simple_ops = {
        .enqueue                = (void *)simple_enqueue,
        .init                   = (void *)simple_init,
        .exit                   = (void *)simple_exit,
        .name                   = "simple",
};
```

Use BPF struct_ops

Thanks