mass attachment of tracing probes

jiri olsa

mass attach

- batch attach support
- for trampolines

what’s wrong

wildcard attach in bpftrace take forever

```
bpftrace -e ‘kfunc:x86*’ { … }
```

same problem in retsnoop and in any tool that wants to attach MANY trampolines

```
bpftrace -e 'kfunc:x86*'
```


```
bpftrace -e 'kfunc:x86*'


...
kfunc:x86_pmu_enable
kfunc:x86_pmu_enable_all
kfunc:x86_pmu_enable_event
kfunc:x86_pmu_event_idx
kfunc:x86_pmu_event_init
kfunc:x86_pmu_event_mapped
kfunc:x86_pmu_event_unmapped
kfunc:x86_pmu_extra_regs
kfunc:x86_pmu_filter_match
kfunc:x86_pmu_handle_irq
kfunc:x86_pmu_hw_config
kfunc:x86_pmu_max_precise
kfunc:x86_pmu_online_cpu
kfunc:x86_pmu_prepare_cpu
kfunc:x86_pmu_read
kfunc:x86_pmu_sched_task
kfunc:x86_pmu_show_pmu_cap
kfunc:x86_pmu_start
kfunc:x86_pmu_start_txn
kfunc:x86_pmu_starting_cpu
...
```


```
bpftrace -e 'kfunc:x86*'


...
kfunc:x86_pmu_enable
kfunc:x86_pmu_enable_all
                                    t


<x86_pmu_max_precise>:
  NOP
  movzbl 0x1eac98c(%rip),%edx       d
  xor    %eax,%eax                  ped
  mov    %edx,%ecx
  and    $0x18,%ecx                 h
  cmp    $0x8,%cl
  je     <x86_pmu_max_precis..
  retq
                                    _
kfunc:x86_pmu_prepare_cpu
kfunc:x86_pmu_read
kfunc:x86_pmu_sched_task
kfunc:x86_pmu_show_pmu_cap
kfunc:x86_pmu_start
kfunc:x86_pmu_start_txn
kfunc:x86_pmu_starting_cpu
...
```


```
bpftrace -e 'kfunc:x86*'

...
kfunc:x86_pmu_enable
kfunc:x86_pmu_enable_all
kfunc:x86_pmu_event_idx
kfunc:x86_pmu_filter
kfunc:x86_pmu_handle_irq
kfunc:x86_pmu_max_precise
kfunc:x86_pmu_prepare_cpu
kfunc:x86_pmu_read
kfunc:x86_pmu_sched_task
kfunc:x86_pmu_show_pmu_cap
kfunc:x86_pmu_start
kfunc:x86_pmu_start_txn
kfunc:x86_pmu_starting_cpu
...
```

```
<x86_pmu_max_precise>:
  NOP
  movzbl 0x1eac98c(%rip),%edx
  xor    %eax,%eax
  mov    %edx,%ecx
  and    $0x18,%ecx
  cmp    $0x8,%cl
  je     <x86_pmu_max_precis..
  retq
```

```
<x86_pmu_max_precise>:
  CALL   TRAMPOLINE
  movzbl 0x1eac98c(%rip),%edx
  xor    %eax,%eax
  mov    %edx,%ecx
  and    $0x18,%ecx
  cmp    $0x8,%cl
  je     <x86_pmu_max_precis..
  retq
```


- attach layer
  - ftrace direct entries API
  - text_poke_bp
  - both depend on ftrace setup
    - gcc -pg -mfentry

ftrace direct API

```
register_ftrace_direct(ip, addr);

unregister_ftrace_direct(ip, addr);

modify_ftrace_direct(ip, old_addr, new_addr);
```


```
prog = sys_bpf(BPF_PROG_LOAD)
fd = sys_bpf(BPF_RAW_TRACEPOINT_OPEN, prog)
```

user

---

kernel

```
bpf_tracing_prog_attach
  bpf_trampoline_link_prog
    bpf_trampoline_update
    {
      arch_prepare_bpf_trampoline

      register_fentry
        register_ftrace_direct
        {
          register direct entry
          rcu sync timeout
        }
    }
```


```
prog = sys_bpf(BPF_PROG_LOAD)
fd = sys_bpf(BPF_RAW_TRACEPOINT_OPEN, prog)
```

user

---

kernel

```
bpf_tracing_prog_attach
  bpf_trampoline_link_prog
    bpf_trampoline_update
    {
      arch_prepare_bpf_trampoline

      register_fentry
        register_ftrace_direct
        {
          register direct entry
          rcu sync timeout
        }
    }
```

PROBLEM ;-)

new batch attach API

- attach all functions in one call
- BPF and ftrace API

new batch attach - bpf API

```
#define BPF_F_MULTI_FUNC (1U << 5)


struct { /* struct used by BPF_LINK_CREATE command */
  ...
  struct {
    __aligned_u64 btf_ids;     /* addresses to attach */
    __u32         btf_ids_cnt; /* addresses count */
  } multi;
} link_create;

enum bpf_link_type = {
  ...
  BPF_LINK_TYPE_TRACING_MULTI = 8;
};
```


new batch attach - ftrace API

```
struct ftrace_ops ops;

ftrace_set_filter_ip(ops, ip, remove, reset);

register_ftrace_direct_multi(ops, addr);
unregister_ftrace_direct_multi(ops);
modify_ftrace_direct_multi(ops, addr);
```


```
prog = sys_bpf(BPF_PROG_LOAD, BPF_F_MULTI_FUNC)
fd = sys_bpf(BPF_LINK_CREATE, prog, [ip1, ip2, ip3, ... ])
```

user

---

kernel

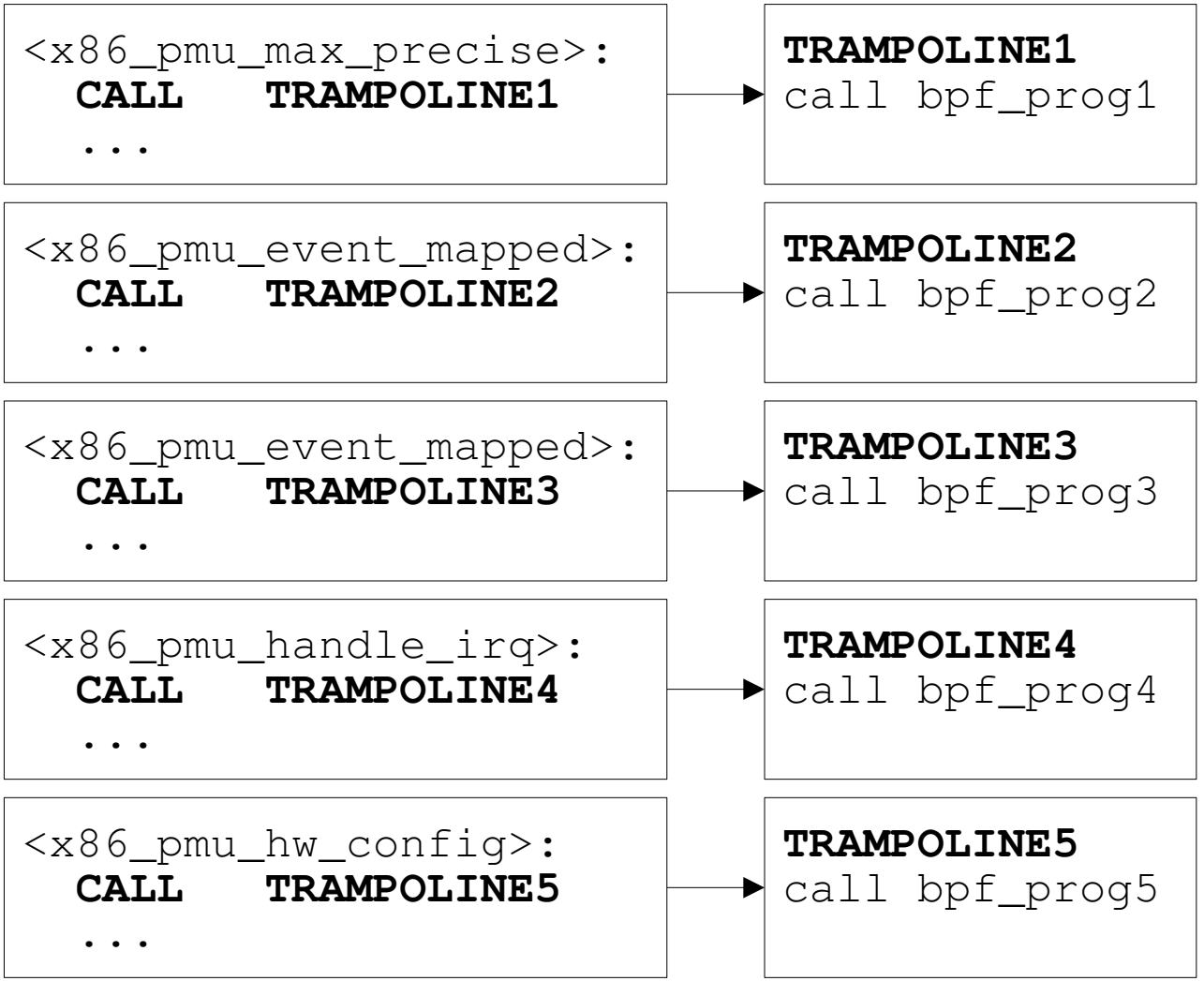
```
link_create
...

    bpf_trampoline_update
    {
      arch_prepare_bpf_trampoline

      register_fentry
        register_ftrace_direct_multi
        {
          register direct entries
          rcu sync timeout
        }
    }
```

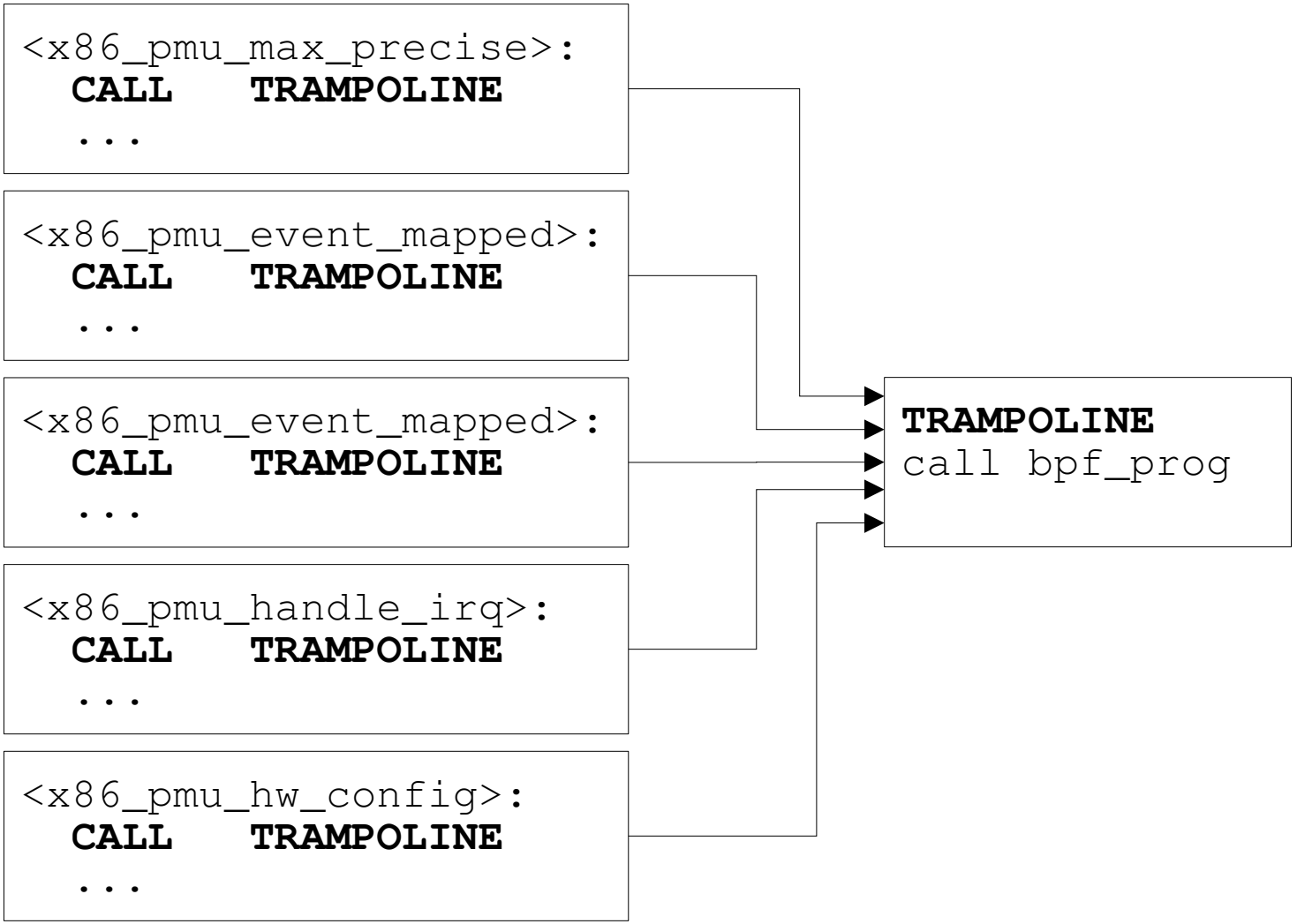

ftrace graph vs direct API issue

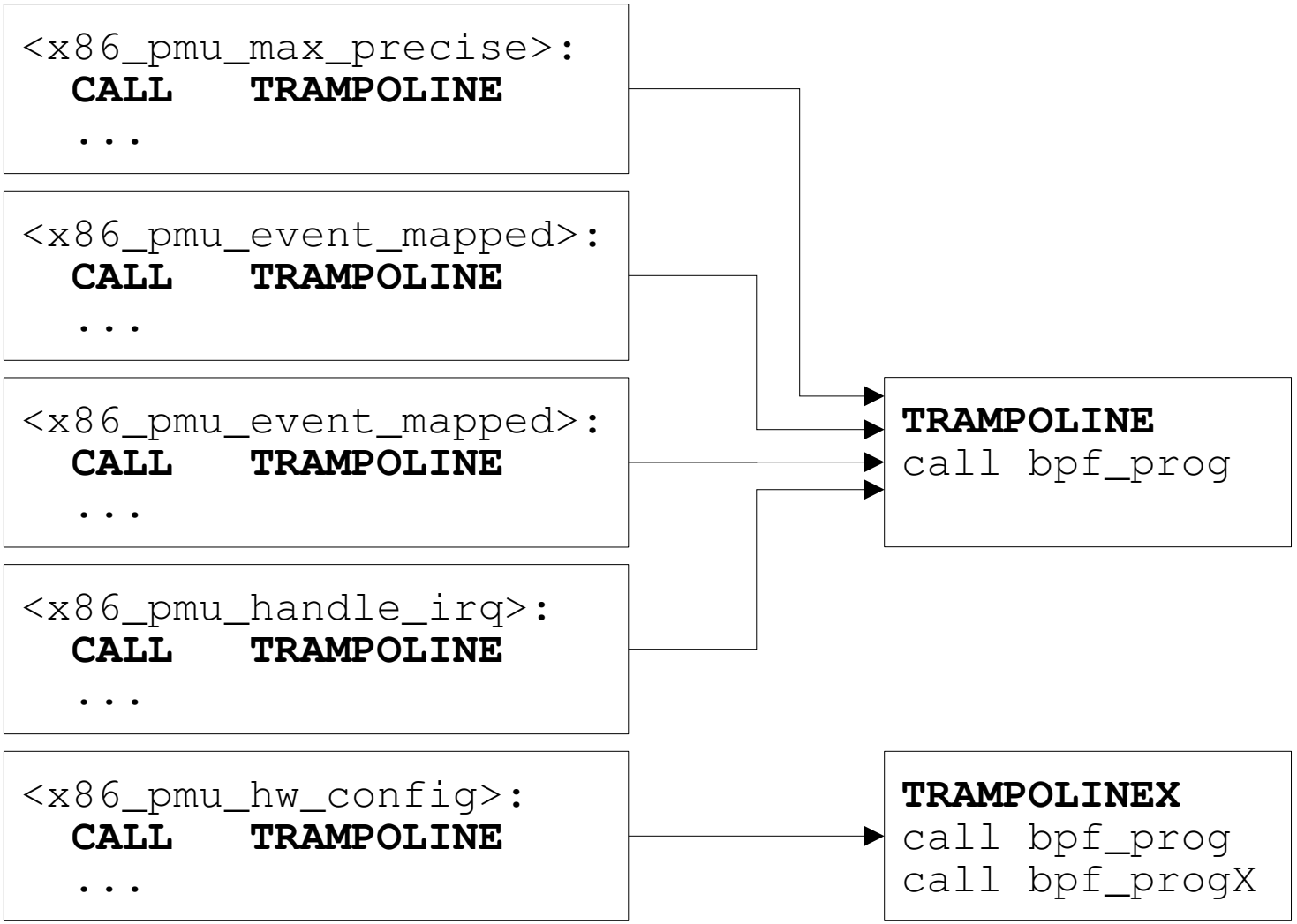
- can’t trace graph with direct entry
- moved direct entry processing into ftrace_ops func

generic trampoline

one trampoline to serve them all


<x86_pmu_max_precise>:
CALL TRAMPOLINE1
...
TRAMPOLINE1
call bpf_prog1
<x86_pmu_event_mapped>:
CALL TRAMPOLINE2
...
TRAMPOLINE2
call bpf_prog2
<x86_pmu_event_mapped>:
CALL TRAMPOLINE3
...
TRAMPOLINE3
call bpf_prog3
<x86_pmu_handle_irq>:
CALL TRAMPOLINE4
...
TRAMPOLINE4
call bpf_prog4
<x86_pmu_hw_config>:
CALL TRAMPOLINE5
...
TRAMPOLINE5
call bpf_prog5


<x86_pmu_max_precise>:
CALL TRAMPOLINE
...
<x86_pmu_event_mapped>:
CALL TRAMPOLINE
...
<x86_pmu_event_mapped>:
CALL TRAMPOLINE
...
<x86_pmu_handle_irq>:
CALL TRAMPOLINE
...
<x86_pmu_hw_config>:
CALL TRAMPOLINE
...
TRAMPOLINE
call bpf_prog


<x86_pmu_max_precise>:
CALL TRAMPOLINE
...
<x86_pmu_event_mapped>:
CALL TRAMPOLINE
...
<x86_pmu_event_mapped>:
CALL TRAMPOLINE
...
<x86_pmu_handle_irq>:
CALL TRAMPOLINE
...
<x86_pmu_hw_config>:
CALL TRAMPOLINE
...
TRAMPOLINE
call bpf_prog
TRAMPOLINEX
call bpf_prog
call bpf_progX

generic trampoline

needs to generate/prepare max arguments

```
BPF_PROG(fentry, arg1, arg2, arg.. )
```

has problem with fexit interface

```
BPF_PROG(fexit, arg1, arg2, arg3, arg4, ret)
BPF_PROGX(fexit, arg1, arg2, ret)
```

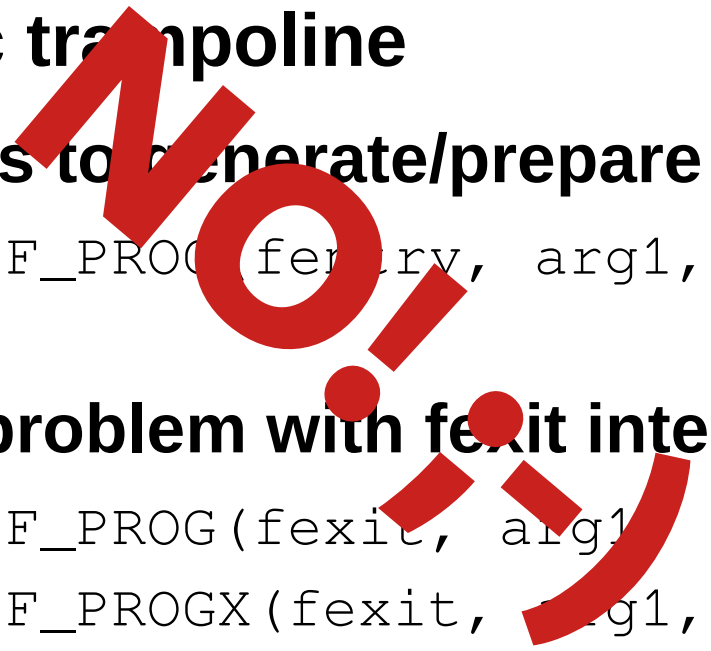

generic trampoline

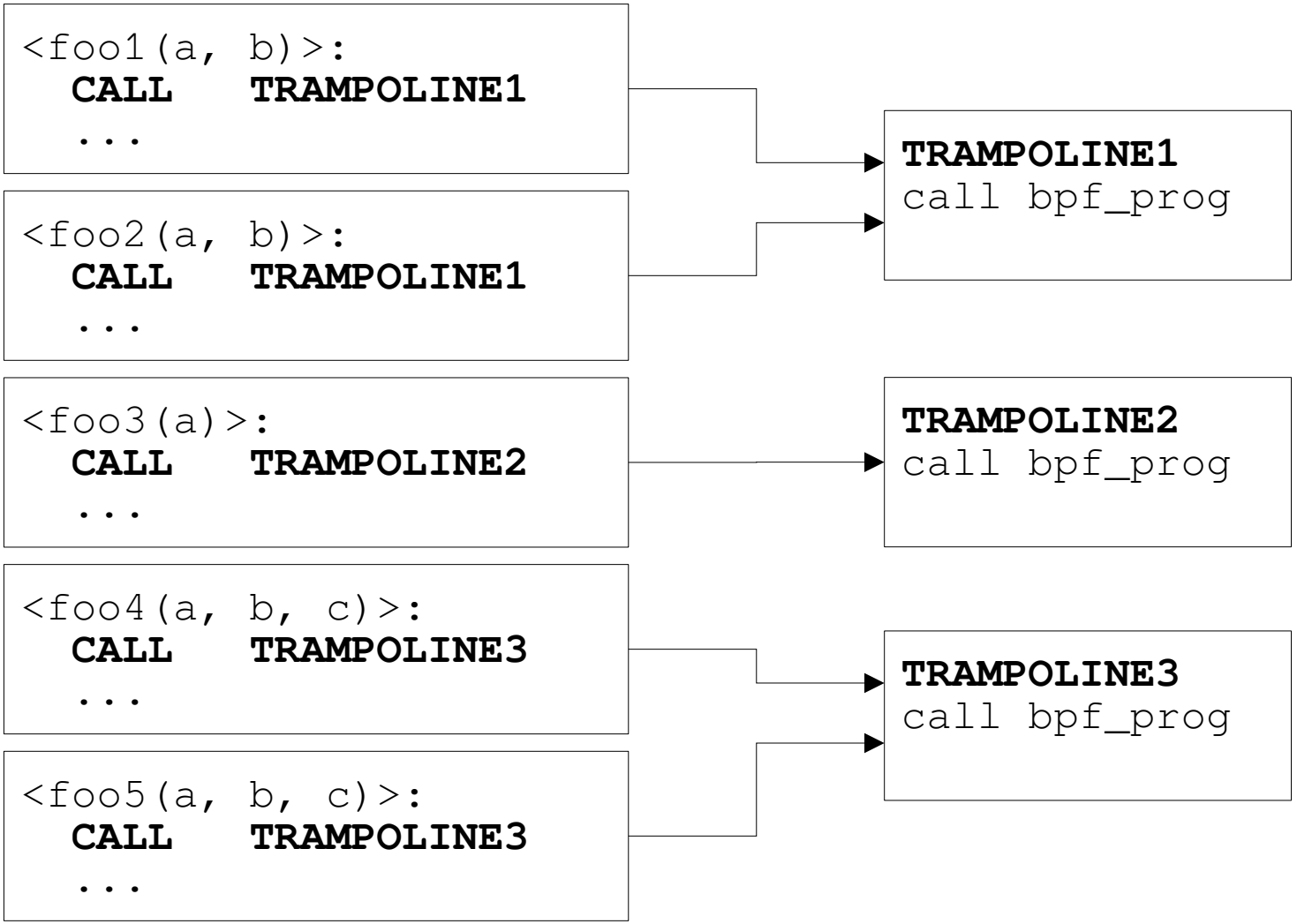
- needs to generate/prepare max arguments

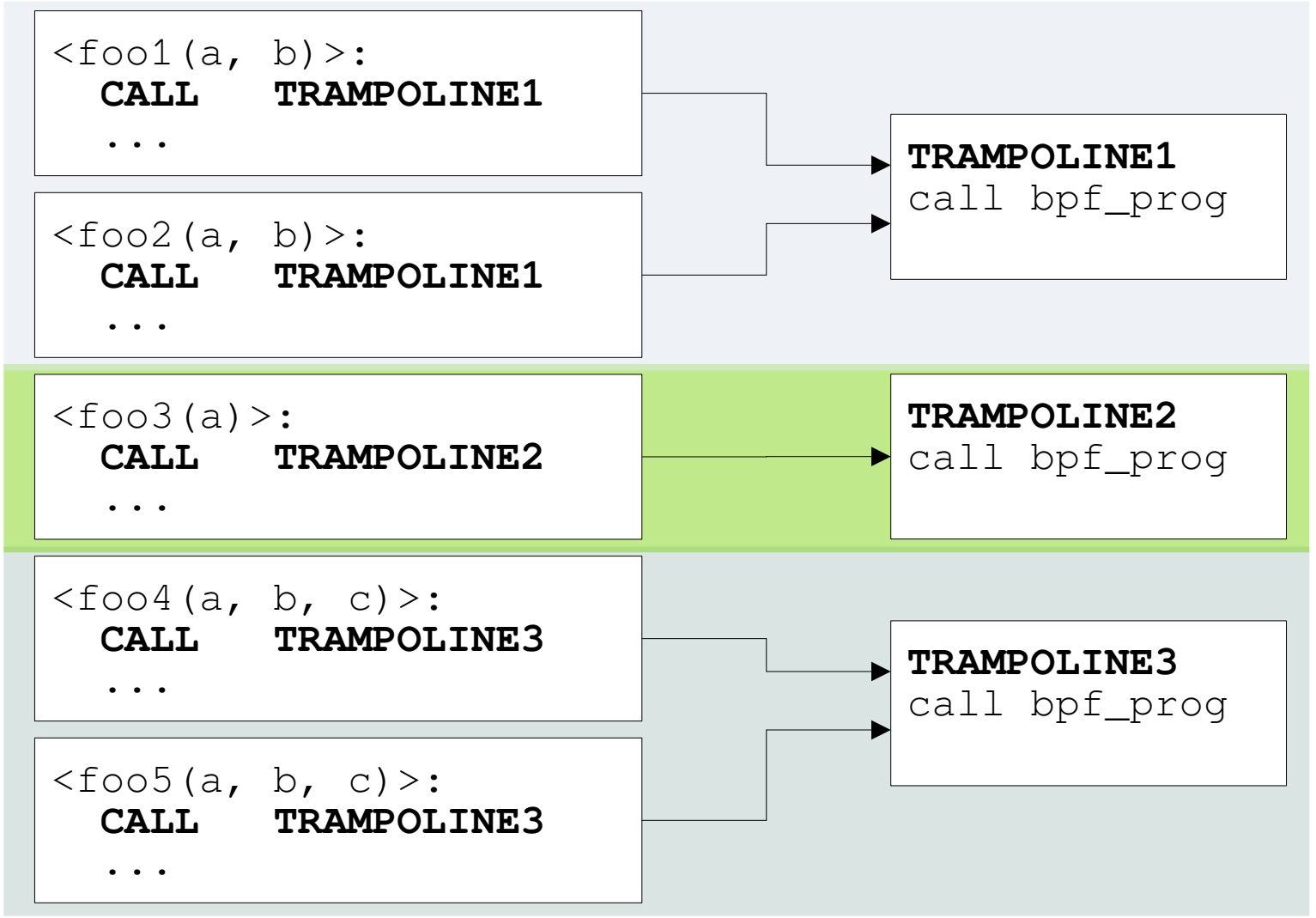
```
BPF_PROG(fentry, arg1, arg2, arg.. )
```

- has problem with fexit interface

```
BPF_PROG(fexit, arg1, arg2, arg3, arg4, ret)
BPF_PROGX(fexit, arg1, arg2, ret)
```



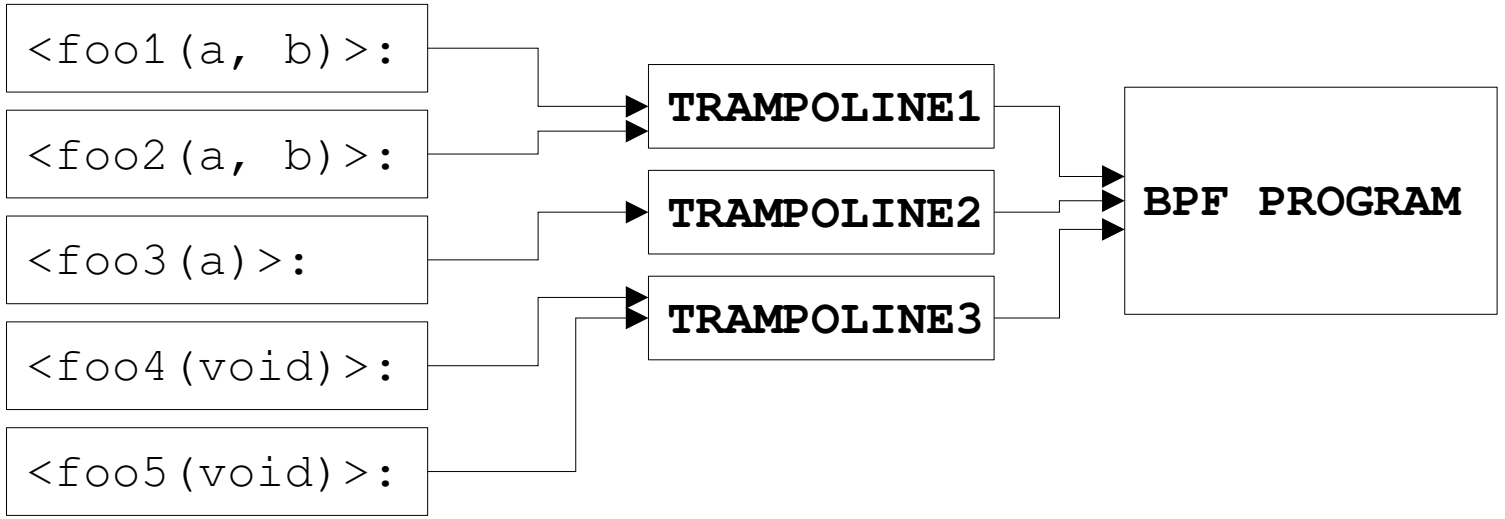

<foo1(a, b)>:
CALL TRAMPOLINE1
...
<foo2(a, b)>:
CALL TRAMPOLINE1
...
<foo3(a)>:
CALL TRAMPOLINE2
...
<foo4(a, b, c)>:
CALL TRAMPOLINE3
...
<foo5(a, b, c)>:
CALL TRAMPOLINE3
...
TRAMPOLINE1
call bpf_prog
TRAMPOLINE2
call bpf_prog
TRAMPOLINE3
call bpf_prog


<foo1(a, b)>:
CALL TRAMPOLINE1
...
<foo2(a, b)>:
CALL TRAMPOLINE1
...
<foo3(a)>:
CALL TRAMPOLINE2
...
<foo4(a, b, c)>:
CALL TRAMPOLINE3
...
<foo5(a, b, c)>:
CALL TRAMPOLINE3
...
TRAMPOLINE1
call bpf_prog
TRAMPOLINE2
call bpf_prog
TRAMPOLINE3
call bpf_prog

new helpers

get traced function ip

get arguments



bpf_get_func_ip helper

already merged

u64 bpf_get_func_ip(void *ctx)

```
Description
        Get address of the traced function
        (for tracing and kprobe programs).
Return
        Address of the traced function.
```


bpf_arg / bpf_ret_value helpers

not posted yet

u64 bpf_arg(void *ctx, int n)

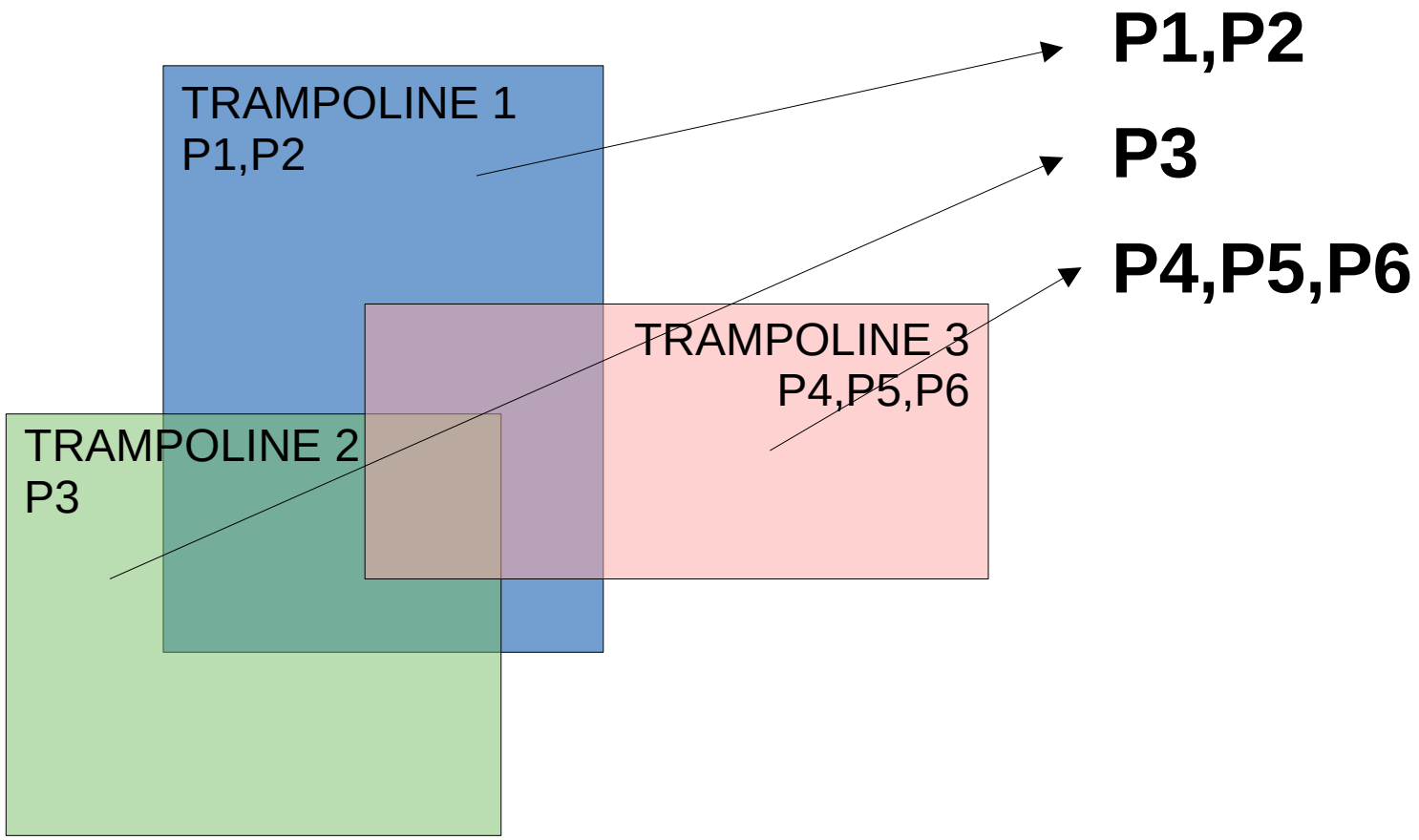
```
Description
        Get n-th argument of the traced function
        (for tracing programs).
Return
        Value of the argument.
```

u64 bpf_ret_value(void *ctx)

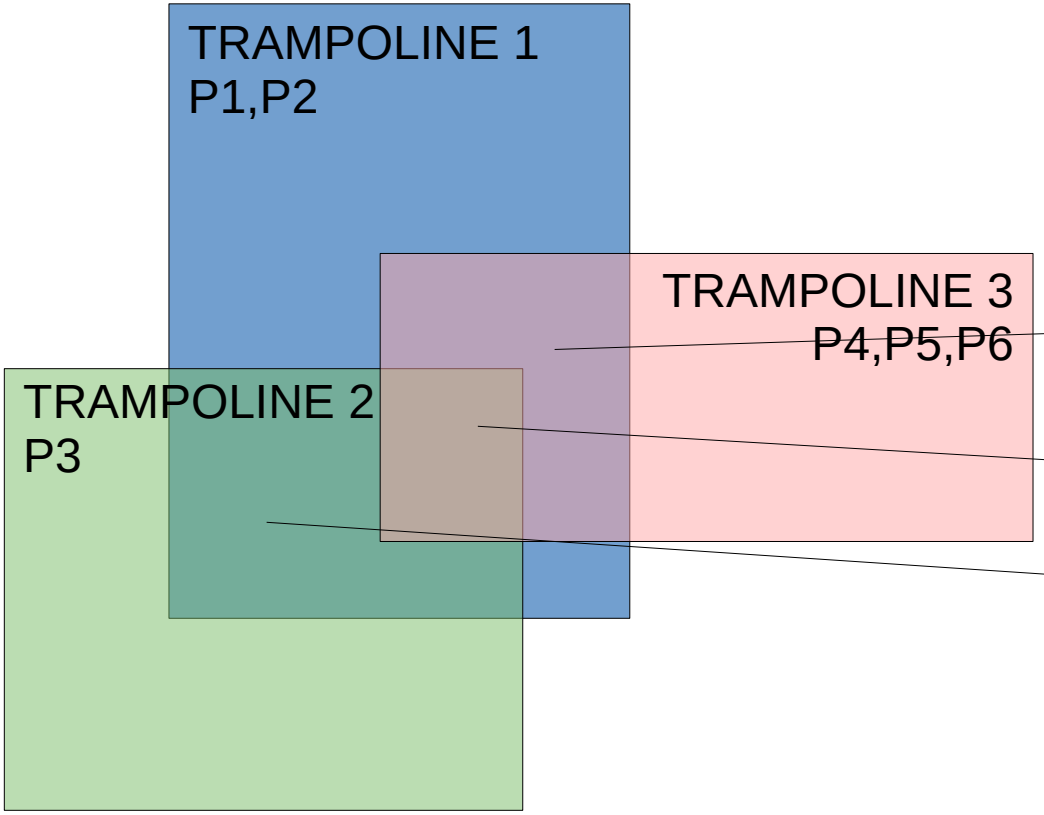
```
Description
        Get return value of the traced function
        (for tracing programs).
Return
        Return value of the traced function.
```


mixing trampolines

mixing trampolines



mixing trampolines



P1,P2

P3

P4,P5,P6

P1,P2,P4,P5,P6

P1,P2,P3,P4,P5,P6

P1,P2,P3

current status

- ftrace direct vs graph tracer
- ftrace direct batch API
- BPF batch attach support
- 2 versions posted, 3rd in progress

thanks, questions..