Google

Kernel

# Use ftrace_regs for function tracing

Simplify kernel interface

LPC23 - Networking & BPF summit

Masami Hiramatsu (Google) <mhiramat@kernel.org>

## Self Introduction

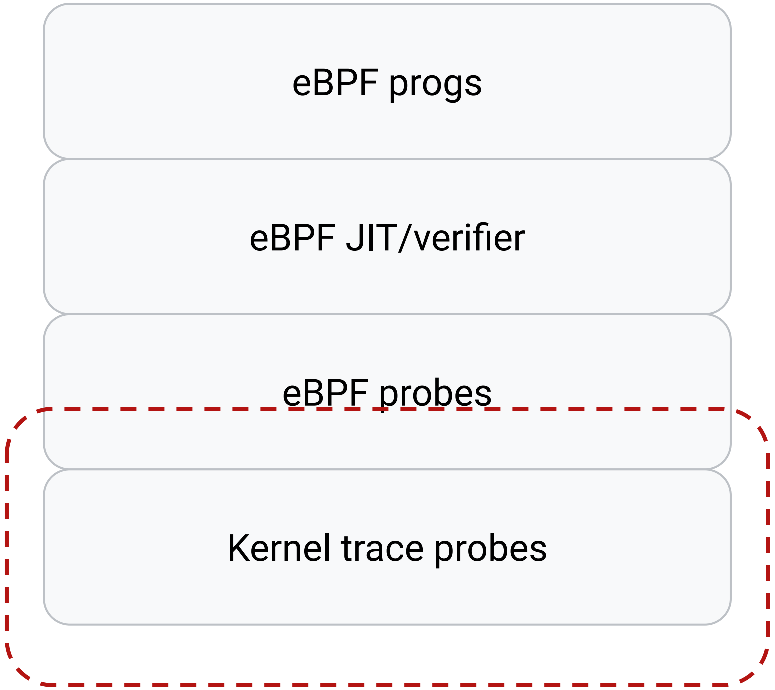
Masami Hiramatsu

- Co-maintainer of tracing tree.
- Maintainer of the *probes and bootconfig
    - And others x86 instruction decoder, perf- probe etc.
- Working for Chrome OS platform

## Introduction

## This talk is about the kernel tracing layer

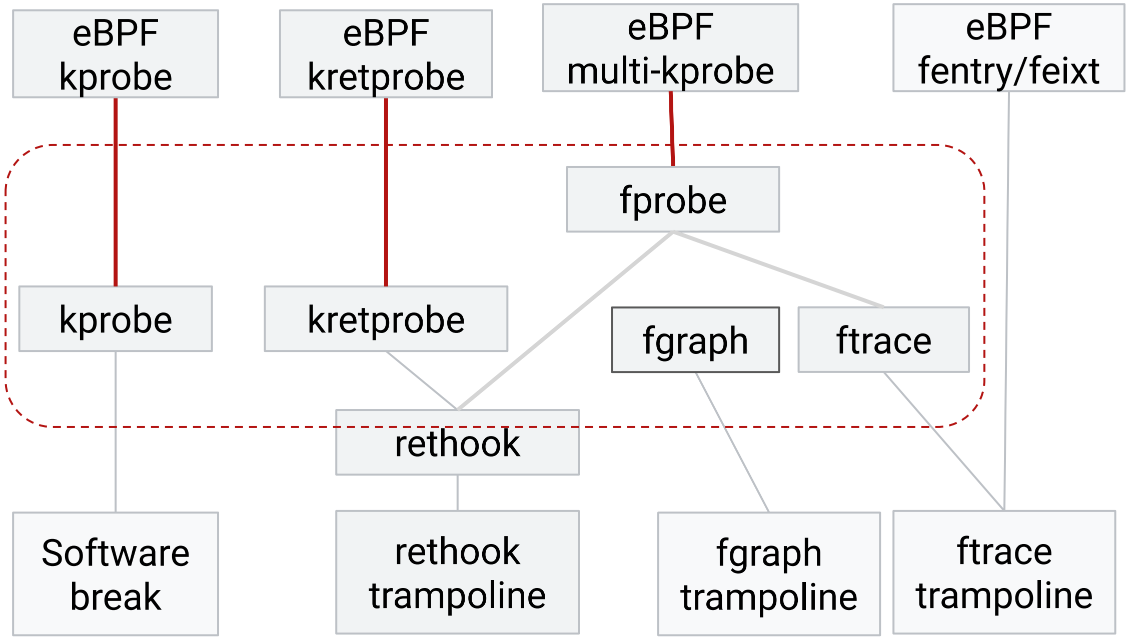
This talk will focus on the tracing backend of eBPF, no eBPF application, nor JIT (but related to JIT)



# Current eBPF trace-side layers (for kernel tracing)

**eBPF** supports kprobe, kretprobe, multi-kprobe/kretprobe, fentry/fexit probes. These are using kprobe, kretprobe, **fprobe** and ftrace trampoline directly.

Today's talk focuses on ***probe** interfaces.



## Pt_regs for tracers

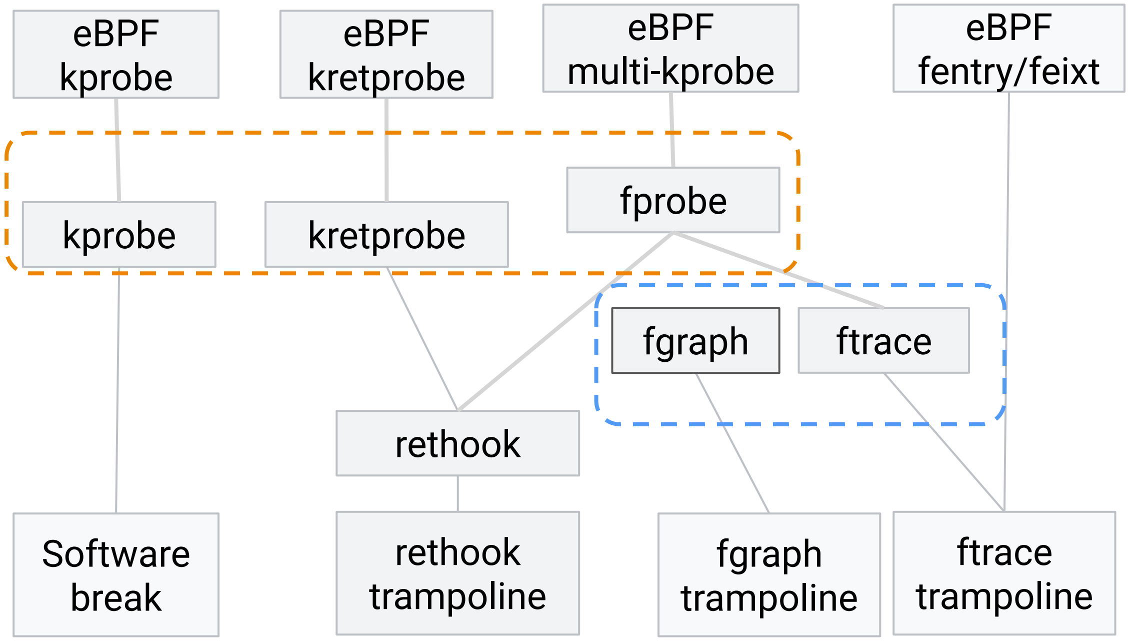
# Current usage of pt_regs in tracers

Current state of pt_regs users

- kprobe-event, uprobe-event, fprobe
- eBPF - kprobe, kretprobe, multi-kprobe/kretprobe, uprobe, USDT

And ftrace_regs users

- Ftrace
- Function graph tracer (internally)



Google

# What is pt_regs?

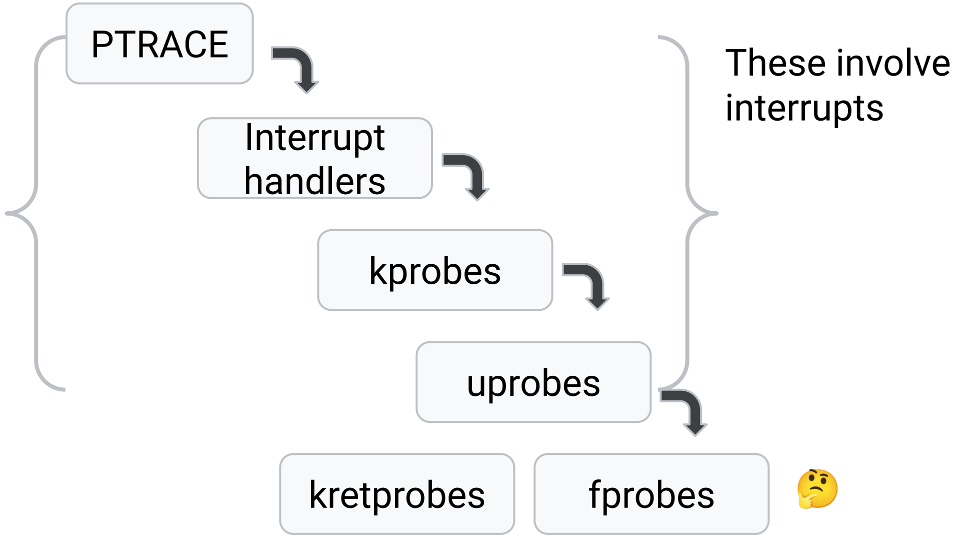
Pt_regs means "ptrace registers"

- Introduced for abstracting registers for ptrace syscall
- Save **all registers** at interrupt

Used in the interrupt handlers

- And reused by kprobe and kretprobe
- And reused by uprobe
- And reused by fprobe...?

Fprobe doesn't use any interrupt.
(kretprobe depends on architecture)



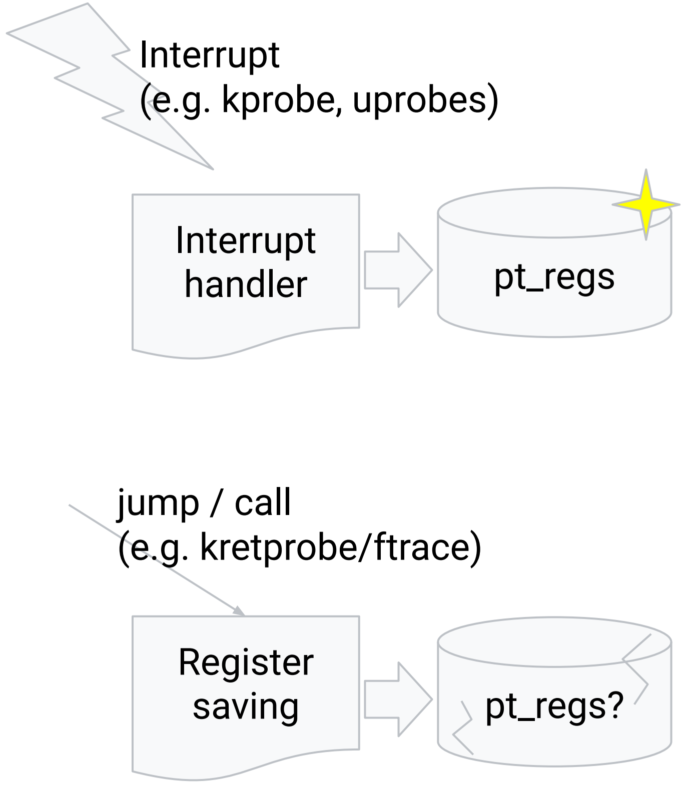
# Problem of using pt_regs in non interrupt context

**pt_regs** is designed for storing all registers in the interrupt context (some registers are saved automatically)

- Some **registers can not be saved** manually (e.g. pstate @arm64)
- Most of the registers are not used but **take time to save** it.

This means, **pt_regs is not correct** and **takes more overhead** if saved manually.

This is the reason why **arm64 doesn't support kprobes on ftrace and rethook**. (and it should not support kretprobe too)



# Current parameters for kernel tracers

There are three tracers for function entry/exit. But interfaces are different.

Function-graph-tracer

- Entry: **ftrace_regs**
- Exit: fgraph_ret_regs

Fprobe (rethook)

- Entry: (incomplete) pt_regs
- Exit: (incomplete) pt_regs

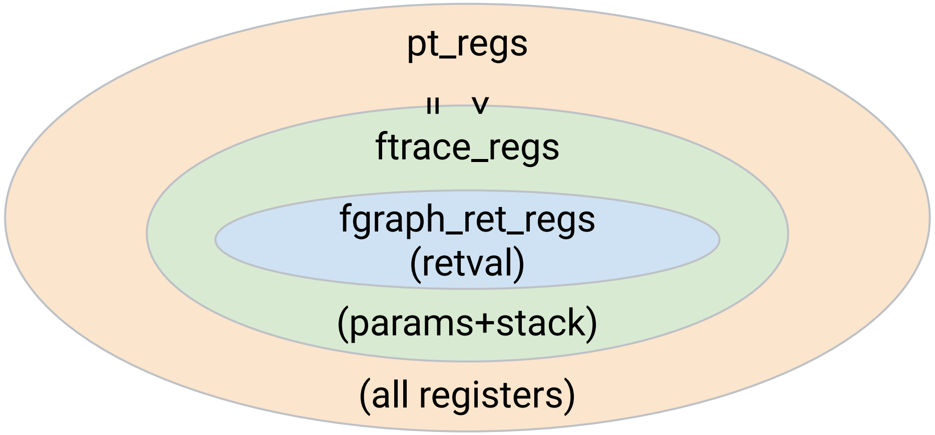
Kprobe/kretprobe

- Entry: pt_regs
- Exit: (incomplete) pt_regs

# Ftrace_regs is a handy option

Ftrace_regs is a partial set of pt_regs (most architectures just wraps pt_regs).

fgraph_ret_regs is a shrunken version of ftrace_regs, but it only has return value.



Google

## What is the ftrace_regs?

ftrace_regs only saves the registers for;

- Function **parameters**
- Function **return values**
- Hooking/unwinding **function call** (e.g. frame pointer, link register or stack pointer and instruction pointer)
- (optional) arch implementation dependent

Don't include state flags, callee-save registers etc.

```
int function_foo(int param1, long param2, void *param3)
{
        ...

        return ret;
}
```

```
[   2.794307]  function_graph_enter_regs+0x184/0x280
[   2.796119]  ? fprobe_selftest_target+0x4/0x20
[   2.797809]  ? test_fprobe_entry+0x91/0x300
[   2.799409]  ? fprobe_selftest_target+0x4/0x20
[   2.801105]  ftrace_graph_func+0xcd/0x170
....
```

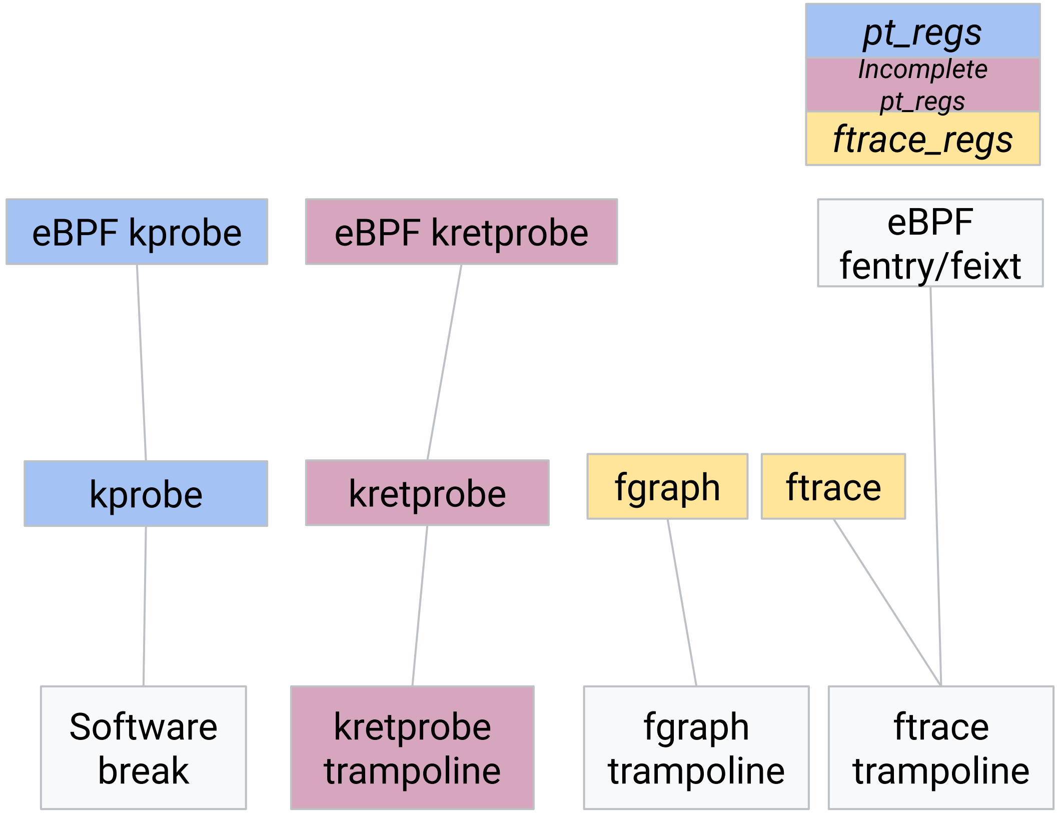
Google

## Kernel tracing changes

# Previous

There were only eBPF kprobe/kretprobe, kfunc/kretfunc.

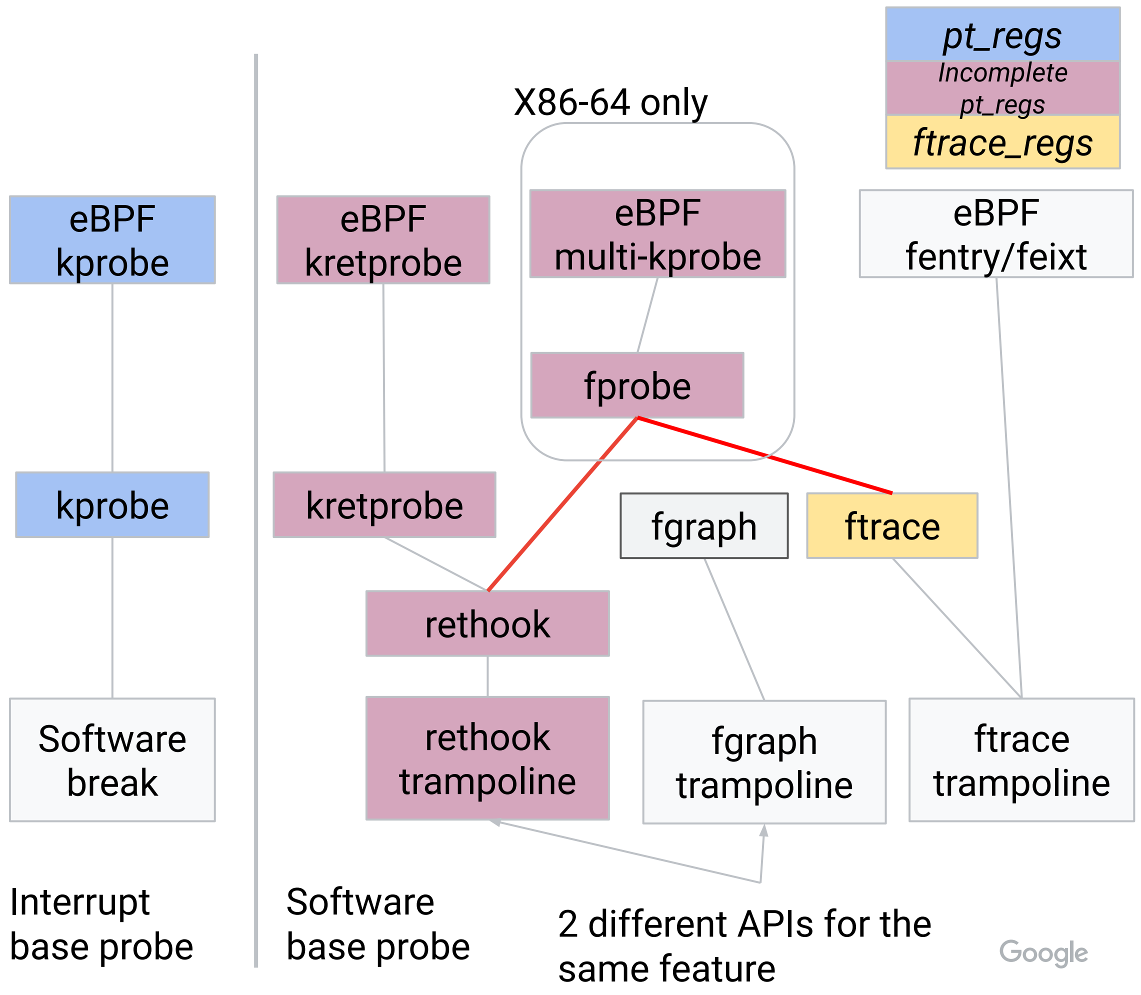
But as you can see, the eBPF kretprobe is working on **incomplete pt_regs**.



Google

## Current

**Fprobe** has been introduced for eBPF multi-kprobe/kretprobe.
But fprobe is based on **ftrace** and **rethook** which provides **ftrace_regs** and **incomplete pt_regs**.

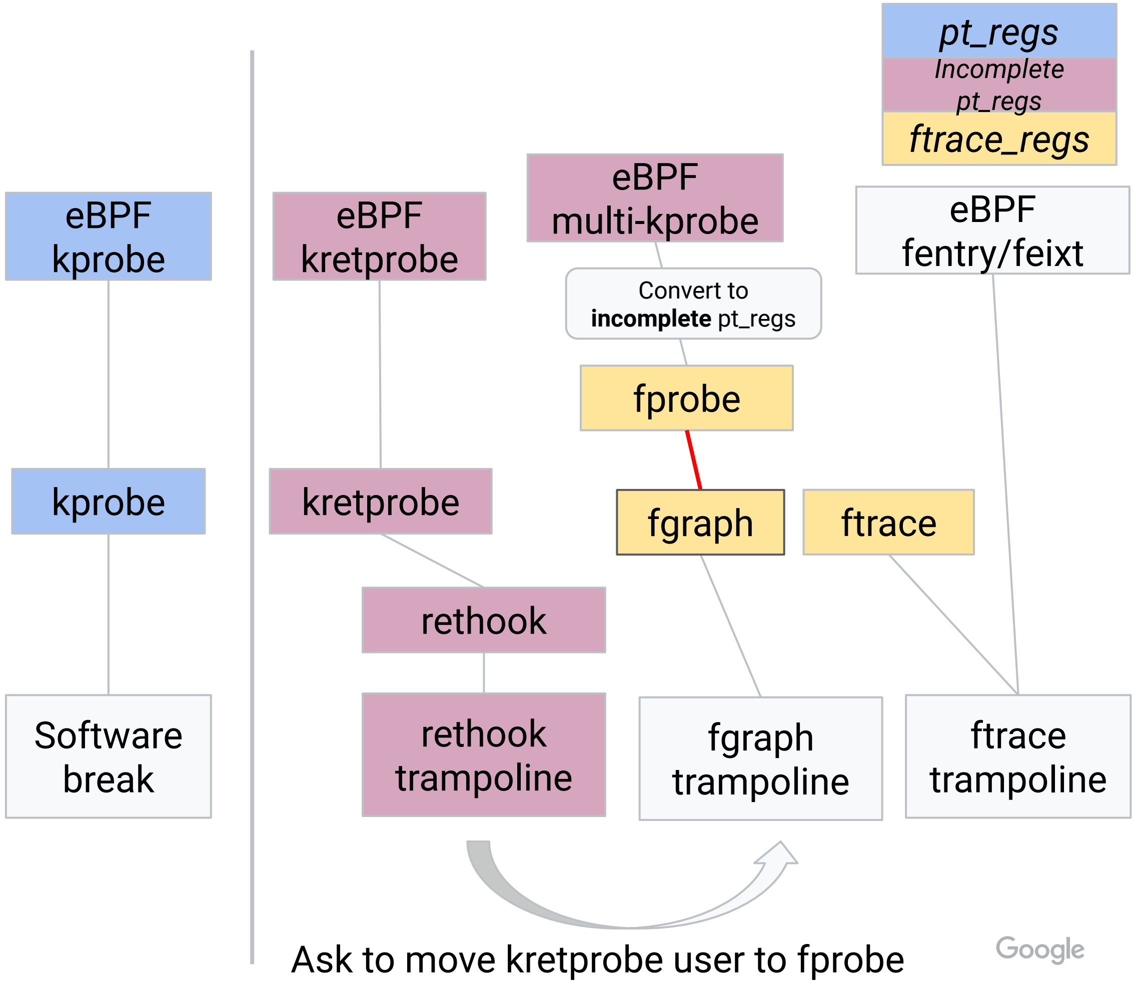


Google

# Next (ongoing) plan

(1) Make func-graph use ftrace_regs
(2) Move fprobe on the func-graph
(3) Convert ftrace_regs to **incomplete pt_regs for eBPF**

Function entry/exit will use ftrace_regs in general.
But eBPF still use incomplete pt_regs.

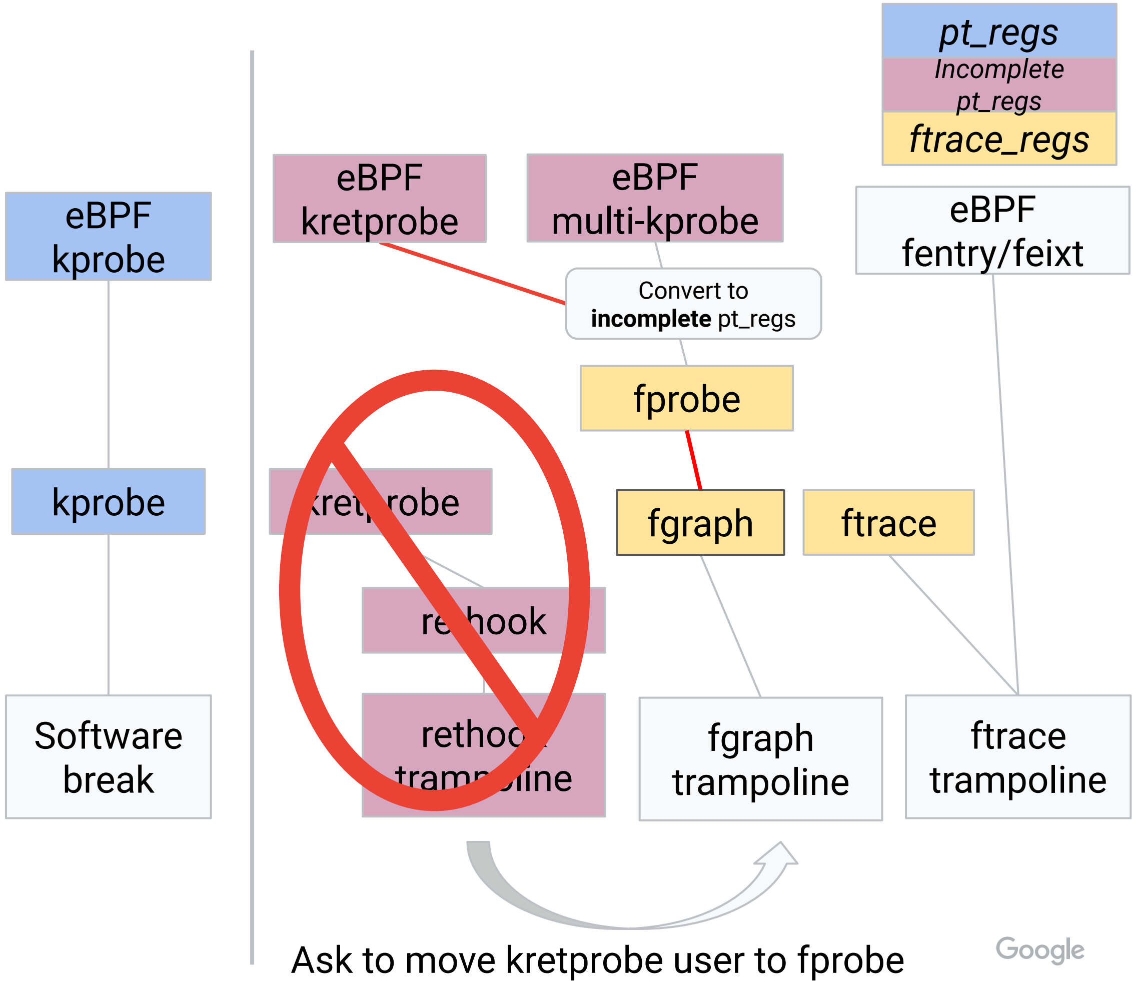


Google

# Next (ongoing) plan

After moving on to the fgraph, I would like to ask kretprobe user to fprobe too, because those have the same function.

For example, eBPF kretprobe also can move onto the fprobe.
Then we can deprecate the kretprobe someday.

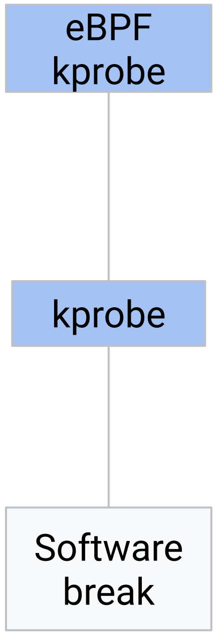


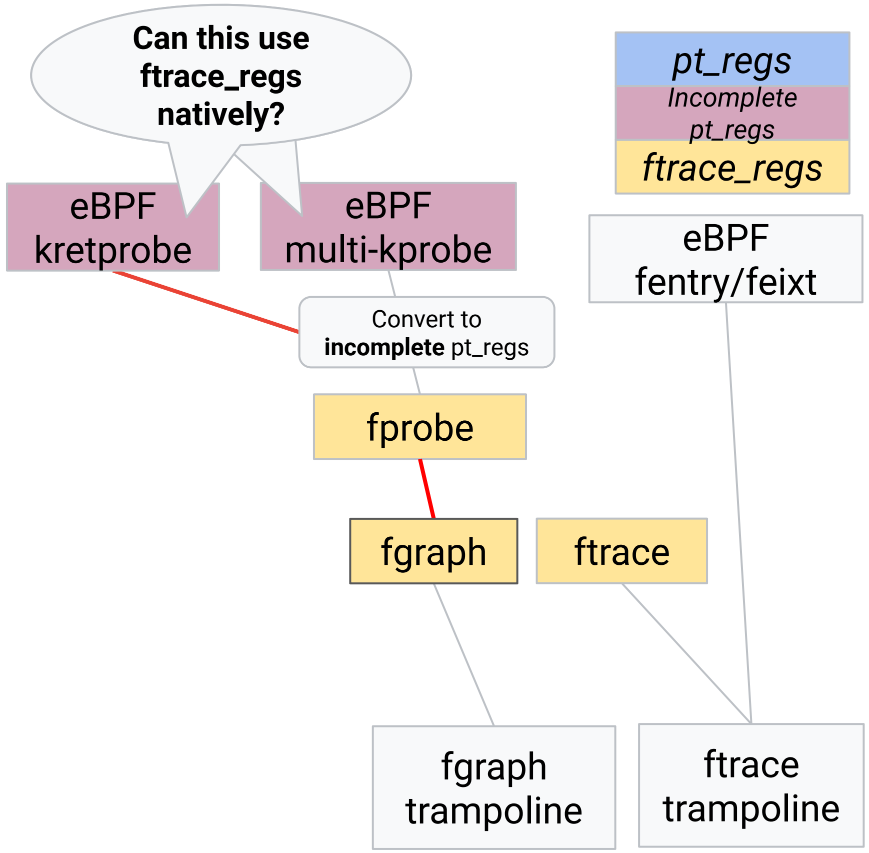
Google

# Incomplete pt_regs exists

However, eBPF is still using the **incomplete pt_regs**.

Can eBPF use ftrace_regs for function entry and exit probes natively?



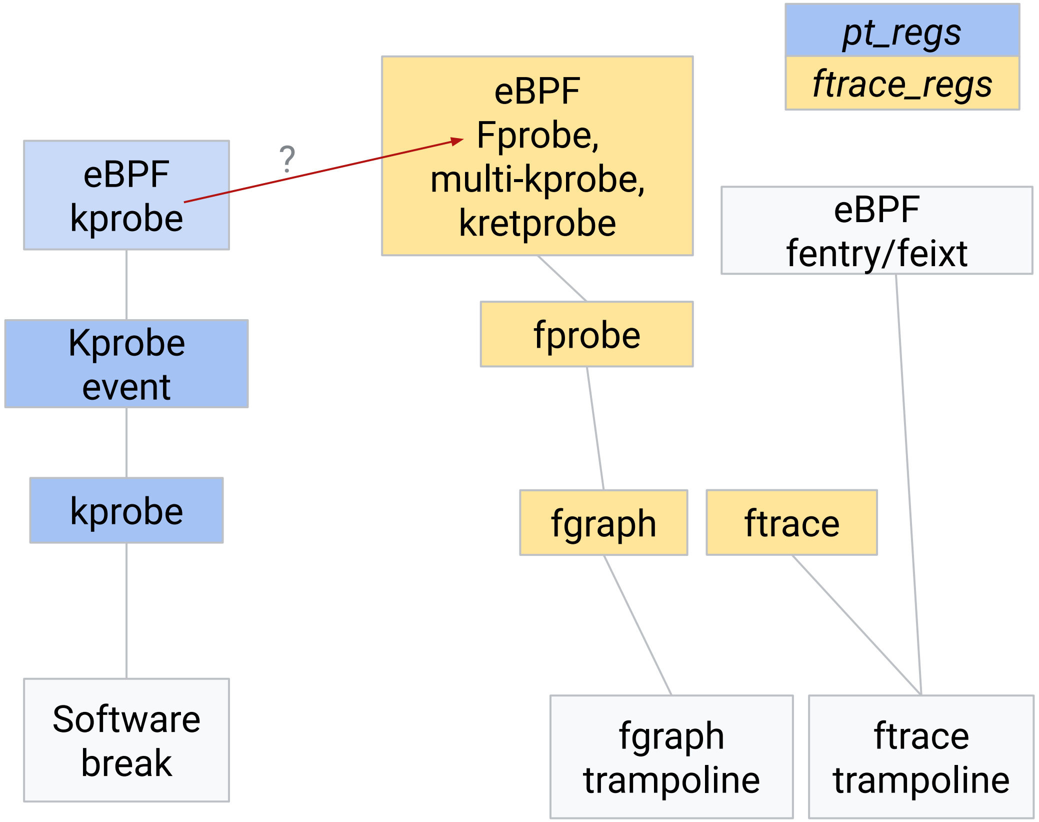


## Future proposal

What about introducing **eBPF fprobe** and using it in addition to eBPF multi-kprobe? (because currently eBPF kprobe **only supports function entry/exit**)

And can it change to use ftrace_regs natively?

If eBPF “kprobe” tries to probe function **BODY**, I would appreciate to help!



## Links:

Fprobe on function graph series (RFC v2)

- Link: https://lore.kernel.org/lkml/169945348320.55307.17578137376868880969.stgit@devnote2/T/

Ftrace_regs discussion

- Link: https://lore.kernel.org/all/20230929102115.09c015b9af03e188f1fbb25c@kernel.org/T/

## Questions?

Thank you!