Google

Probes in the kernel

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

Agenda

01 Probes in the kernel

02 Probes

03 Ftrace features

04 Next Steps

05 Q&A

Probes in the kernel

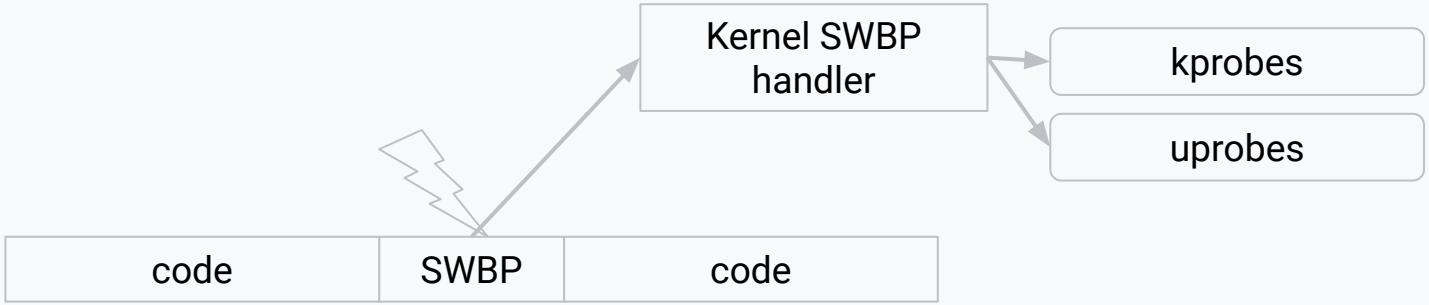
Linux kernel supports many “probes”

- Breakpoint probes : kprobes and uprobes
  - Use breakpoint to probe
- Wrapper probe: fprobe
  - Wrapping function/function graph tracer
- Probes in tracefs: eprobe, tprobe
  - Implemented in tracing (not kernel API)

Kprobes / Uprobes

Set up a **software breakpoint** on target address.
User can insert event almost everywhere.

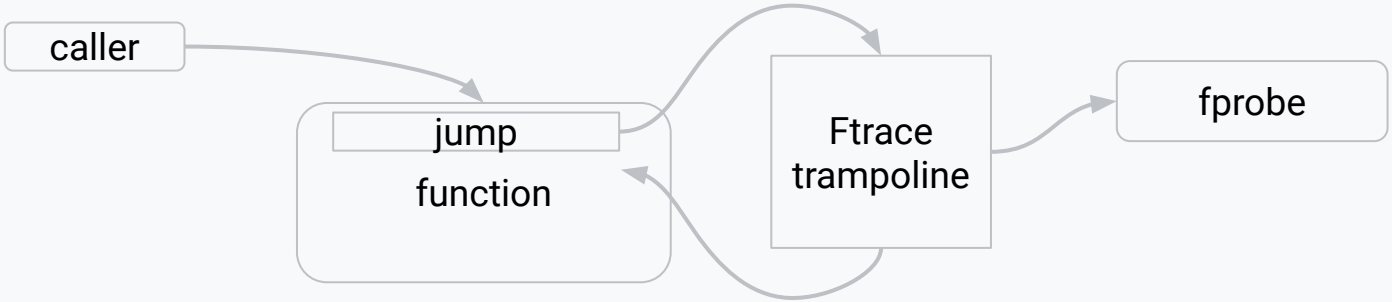
- eBPF / ftrace / perf will use these probes.
- `perf probe` can analyze debuginfo and helps user to probe **source code level**. (function body)
- Kretprobe/Uretprobe can hook the function return.
- Kprobes may optimize SWBP with a jump or ftrace (SWBP:~500ns -> Jump: ~100ns)



Fprobe

Set up **function-tracer** on target **kernel function** to trace function entry and exit.

- Faster registration for multiple places.
  - eBPF kprobe_multi uses this probe.
- BTF(BPF type format) allows user to trace function parameters by name.

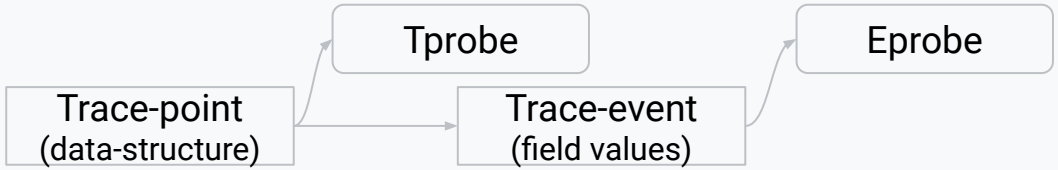


Eprobe / Tprobe

Set up probes on existing events and tracepoints.

- Eprobe: Event probe - only on the trace event
  - Add an eprobe on another **static trace-event** to operate **trace event parameters**.
    - Dereference pointer / change parameter types
  - Parameters needs to be checked via `events/*/*/format`

- Tprobe: Tracepoint probe - only on the tracepoint (!= trace event)
  - Add a tprobe on a **raw tracepoint** to dereference pointer
    - Tracing different fields of given pointers
  - Internally, this is an fprobe variant. Hook the tracepoint by stub function.



Ftrace feature: BTF support

If BTF (BPF Type Format) is supported, user can specify the function parameters and fields of data structure by its **name** and **dot/allow** (./->) operations.

- This feature is enabled on **kprobes** (if it is on function entry/exit), **fprobe** and **tprobe**.
- If **return value** is a pointer of data structure, it can dereference fields by "->".

```
# echo 'f vfs_open path->dentry->d_iname:string path->mnt->mnt_flags:x32 file->f_mode' >>
dynamic_events
# echo 1 > events/fprobes/vfs_open__entry/enable
# tail -n 1 trace
           tail-468     [013] .....   787.122923: vfs_open__entry: (vfs_open+
0x4/0xe0) arg1="trace" arg2=0x10000020 arg3=1
```

Ftrace feature: Entry data support

Function exit probes (kprobe and fprobe) supports **entry data access**.

- If user specifies entry parameter at function exit probe, it is saved at function entry.
  - E.g. checking a data structure field modification by one probe

```
# echo 'f tracefs_create_file%return name:string parent->d_iname:string $retval->d_iname:string
$retval->d_inode->i_uid' >> dynamic_events
# echo 1 > events/fprobes/tracefs_create_file__exit/enable
# mkdir instances/foo
# tail -n 1 trace
           mkdir-463     [010] ...1.   492.968943: tracefs_create_file__exit:
(add_tracer_options+0x220/0x2d0 <- tracefs_create_file) name="test_nop_refuse"
 arg2="options" arg3="test_nop_refuse" arg4=0x0
```

What’s next?

- HWBP probe for tracing a specific variable access
- Monitor probe for periodically monitoring kernel statistics
- Any ideas?

Q & A