



Contribution ID: 124

Type: **not specified**

## Make `ftrace_regs` a common trace interface for function entry/exit tracing

*Tuesday, 14 November 2023 09:30 (30 minutes)*

We are looking for the new register-set data structure, instead of `pt_regs`, for function entry/exit trace events. This is because `pt_regs` is expected to save all registers including some control registers which are usually saved when an exception or interrupt happens. However, using `ftrace` it will not be able to be used on some architecture. Moreover, for most RISC architecture, saving all registers will take a lot of time and consume a large amount of stacks. And that is useless on function entry and exit since the registers which we need are a part of registers which can be used for passing function parameters, or return value and stacks.

Previously, we have only `kprobe` which uses `pt_regs` because it is based on the software breakpoint, which is usually implemented as an exception and saves `pt_regs` automatically.

Now, we have `fprobe` for function entry/exit tracing, which is based on `ftrace` and `rethook`.

From the `tracefs` user's point of view, `fprobe` is used for `fprobe-event`. And users are only able to access function arguments and function return value, and stacks from the `fprobe` event. Thus we don't need to use `pt_regs`.

The problem is that the `eBPF`. Since `fprobe` is used for `eBPF` to enable multiple `kprobe` events, which expects the handler will access registers via `pt_regs` data structure (but usually only access limited registers for function arguments). So it can be updated to `ftrace_regs` too, but needs another interface.

Once we moved to `ftrace_regs`, we can start integrating `rethook` with function-graph tracer. Both implement the shadow stack but in different ways. If those provide the same interface, we can choose one of them.

**Primary author:** HIRAMATSU, Masami (Google)

**Presenter:** HIRAMATSU, Masami (Google)

**Session Classification:** eBPF & Networking

**Track Classification:** eBPF & Networking Track