Function Parameters with BTF

Function tracing with parameters

Mr Masami Hiramatsu (Google)
Steven Rostedt (Google)

Function graph with return values

CONFIG_FUNCTION_GRAPH_RETVAL will enables function graph tracer to show the function return values.

But this doesn't check the function return value type.

```
tracing # echo 1 > options/funcgraph-retval

tracing # echo function_graph > current_tracer

tracing # cat trace
...
 0)   0.262 us    |              do_raw_spin_unlock(); /*
= 0x0 */
 0)   0.242 us    |              preempt_count_sub(); /*
= 0x80000001 */
 0)   1.133 us    |            } /*
_raw_spin_unlock_irqrestore = 0x80000000 */
 0)               |            down_read() {
 0)               |              stack_trace_save() {
 0)               |                arch_stack_walk() {
 0)               |                  __unwind_start() {
 0)   0.189 us    |                    get_stack_info();
/* = 0x0 */
 0)               |
unwind_next_frame() {
```

Function graph with kernel parameters

Since function graph entry handler saves **ftrace_regs**, it can get the function parameter values.

But it doesn’t know the number of kernel parameters and its types. -> **Use BTF**

BTF function prototype information

2.2.13 BTF_KIND_FUNC_PROTO defines the "function prototype" type, which includes the type of

- Return value
- Function parameters

And the number of parameters.

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

        return ret;
}
```

- # of params = 3
- Type of 1st param = 32bit signed integer
- Type of 2nd param = 64bit signed integer
- Type of 3rd param = pointer to void
- Return type is 32bit signed integer

Function graph tracer with BTF

BTF will tell us the function prototypes!

- Showing all parameters (or first N parameters)
- Checking function return value type (e.g. skip void functions)

Discussions

E.g.

- How many parameters are saved?
- Performance degradation?
- Interface?
- Not only raw parameters?

Appendix

BTF - BPF Type Format

BTF stores the “type” information in C language into the kernel binary.

This information is for each functions (except for inlined function). IOW, most of symbols in kallsyms.

```
struct btf_type {
        __u32 name_off;
        /* "info" bits arrangement
         * bits  0-15: vlen (e.g. # of struct's members)
         * bits 16-23: unused
         * bits 24-28: kind (e.g. int, ptr, array...etc)
         * bits 29-30: unused
         * bit     31: kind_flag, currently used by
         *             struct, union, fwd, enum and enum64.
         */
        __u32 info;
        /* "size" is used by INT, ENUM, STRUCT, UNION and
ENUM64.
         * "size" tells the size of the type it is describing.
         *
         * "type" is used by PTR, TYPEDEF, VOLATILE, CONST,
RESTRICT,
         * FUNC, FUNC_PROTO, DECL_TAG and TYPE_TAG.
         * "type" is a type_id referring to another type.
         */
        union {
                __u32 size;
                __u32 type;
        };
};
```

BTF already used in the ftrace (!= function tracer)

CONFIG_FPROBE_EVENTS/KPROBE_EVENTS and **CONFIG_PROBE_EVENTS_BTF_ARGS** is set, kprobes/kretprobe and function probe events accept function **parameter by name**, and set **correct type to $retval**. (v6.6)

```
tracing # echo 't kfree ptr' >> dynamic_events
```

```
tracing # echo 'f kmem_cache_alloc $arg*' >>
dynamic_events
```

```
tracing # echo 'f kfree $retval' >> dynamic_events
sh: write error: No such file or directory
```

```
tracing # cat error_log
[ 2576.843544] trace_fprobe: error: This function returns
'void' type
  Command: f kfree $retval
                   ^
```

Samples 1: function parameters

```
tracing # echo 'f kmem_cache_alloc $arg*' >> dynamic_events
tracing # cat events/fprobes/kmem_cache_alloc__entry/format
name: kmem_cache_alloc__entry
ID: 1354
...
      field:unsigned long __probe_ip;     offset:8;   size:8;     signed:0;
      field:u64 s;      offset:16;  size:8;     signed:0;
      field:u32 gfpflags;     offset:24;  size:4;     signed:0;

print fmt: "(%lx) s=0x%Lx gfpflags=%u", REC->__probe_ip, REC->s, REC->gfpflags

tracing # echo 1 > events/fprobes/enable
tracing # tail -n 3 trace
            tail-111     [000] .....  2380.500549: kmem_cache_alloc__entry: (kmem_cache_alloc+0x4/0x320)
s=0xffff8880041c2300 gfpflags=3264
            tail-111     [000] .....  2380.500614: kmem_cache_alloc__entry: (kmem_cache_alloc+0x4/0x320)
s=0xffff8880041c2e00 gfpflags=3264
            tail-111     [000] .....  2380.500769: kmem_cache_alloc__entry: (kmem_cache_alloc+0x4/0x320)
s=0xffff888004045b00 gfpflags=3264
```

Samples 2: function return value

```
(void check)
tracing # echo 'f kfree $retval' >> dynamic_events
sh: write error: No such file or directory


tracing # cat error_log
[ 2576.843544] trace_fprobe: error: This function returns 'void' type
  Command: f kfree $retval
                   ^
(subfield)
tracing # echo 'f task_rq_lock cpu=$retval->cpu nr_running=$retval->nr_running' >> dynamic_events
```

Samples 3: trace parameters and subfields

```
tracing # echo 't sched_switch pid=prev->pid comm=prev->comm:string state=prev->__state
saved_state=prev->saved_state' > dynamic_events

tracing # cat events/tracepoints/sched_switch/format
name: sched_switch
ID: 1354
…
      field:unsigned long __probe_ip;     offset:8;   size:8;     signed:0;
      field:s32 pid;    offset:16;  size:4;     signed:1;
      field:__data_loc char[] comm; offset:24;  size:4;     signed:1;
      field:u32 state;  offset:28;  size:4;     signed:0;
      field:u32 saved_state;  offset:36;  size:4;     signed:0;

print fmt: "(%lx) pid=%d comm=\"%s\" state=%u saved_state=%u", REC->__probe_ip, REC->pid, __get_str(comm),
REC->state, REC->saved_state

tracing # echo 1 > events/tracepoints/sched_switch/enable
tracing # tail -n 2 trace
          <idle>-0       [000] d..3.   530.539688: sched_switch: (__probestub_sched_switch+0x4/0x10) pid=0
comm="swapper/0" state=0 saved_state=0
     rcu_preempt-17      [000] d..3.   530.539698: sched_switch: (__probestub_sched_switch+0x4/0x10) pid=17
comm="rcu_preempt" state=1026 saved_state=0
```