Kprobe-based Event Tracing

Documentation is written by Masami Hiramatsu

Overview

These events are similar to tracepoint based events. Instead of Tracepoint, this is based on kprobes (kprobe and kretprobe). So it can probe wherever kprobes can probe (this means, all functions body except for __kprobes functions). Unlike the Tracepoint based event, this can be added and removed dynamically, on the fly.

To enable this feature, build your kernel with CONFIG_KPROBE_TRACING=y.

Similar to the events tracer, this doesn't need to be activated via current_tracer. Instead of that, add probe points via /sys/kernel/debug/tracing/kprobe_events, and enable it via /sys/kernel/debug/tracing/events/kprobes/<EVENT>/enabled.

Synopsis of kprobe_events

```
  p[:[GRP/]EVENT] SYMBOL[+offs]|MEMADDR [FETCHARGS]     : Set a probe
  r[:[GRP/]EVENT] SYMBOL[+0] [FETCHARGS]                : Set a return probe
  -:[GRP/]EVENT                                         : Clear a probe

 GRP            : Group name. If omitted, use "kprobes" for it.
 EVENT          : Event name. If omitted, the event name is generated
                  based on SYMBOL+offs or MEMADDR.
 SYMBOL[+offs]  : Symbol+offset where the probe is inserted.
 MEMADDR        : Address where the probe is inserted.

 FETCHARGS      : Arguments. Each probe can have up to 128 args.
  %REG          : Fetch register REG
  @ADDR         : Fetch memory at ADDR (ADDR should be in kernel)
  @SYM[+|-offs] : Fetch memory at SYM +|- offs (SYM should be a data symbol)
  $stackN       : Fetch Nth entry of stack (N >= 0)
  $stack        : Fetch stack address.
  $retval       : Fetch return value.(*)
  +|-offs(FETCHARG) : Fetch memory at FETCHARG +|- offs address.(**)
  NAME=FETCHARG : Set NAME as the argument name of FETCHARG.
  FETCHARG:TYPE : Set TYPE as the type of FETCHARG. Currently, basic types
                  (u8/u16/u32/u64/s8/s16/s32/s64) are supported.

  (*) only for return probe.
  (**) this is useful for fetching a field of data structures.
```

Per-Probe Event Filtering

Per-probe event filtering feature allows you to set different filter on each probe and gives you what arguments will be shown in trace buffer. If an event name is specified right after 'p:' or 'r:' in kprobe_events, it adds an event under tracing/events/kprobes/<EVENT>, at the directory you can see 'id', 'enabled', 'format' and 'filter'.

enabled:
  You can enable/disable the probe by writing 1 or 0 on it.

format:
  This shows the format of this probe event.

filter:
  You can write filtering rules of this event.

id:
  This shows the id of this probe event.

Event Profiling

 You can check the total number of probe hits and probe miss-hits via
/sys/kernel/debug/tracing/kprobe_profile.
 The first column is event name, the second is the number of probe hits,
the third is the number of probe miss-hits.

Usage examples

To add a probe as a new event, write a new definition to kprobe_events
as below.

```
  echo 'p:myprobe do_sys_open dfd=%ax filename=%dx flags=%cx mode=+4($stack)' >
/sys/kernel/debug/tracing/kprobe_events
```

 This sets a kprobe on the top of do_sys_open() function with recording
1st to 4th arguments as "myprobe" event. Note, which register/stack entry is
assigned to each function argument depends on arch-specific ABI. If you unsure
the ABI, please try to use probe subcommand of perf-tools (you can find it
under tools/perf/).
As this example shows, users can choose more familiar names for each arguments.

```
  echo 'r:myretprobe do_sys_open $retval' >>
/sys/kernel/debug/tracing/kprobe_events
```

 This sets a kretprobe on the return point of do_sys_open() function with
recording return value as "myretprobe" event.
 You can see the format of these events via
/sys/kernel/debug/tracing/events/kprobes/<EVENT>/format.

```
  cat /sys/kernel/debug/tracing/events/kprobes/myprobe/format
name: myprobe
ID: 780
format:
        field:unsigned short common_type;       offset:0;       size:2;
signed:0;
        field:unsigned char common_flags;       offset:2;       size:1;
signed:0;
        field:unsigned char common_preempt_count;       offset:3;
size:1;signed:0;
```

```
        field:int common_pid;   offset:4;       size:4; signed:1;
        field:int common_lock_depth;    offset:8;       size:4; signed:1;

        field:unsigned long __probe_ip; offset:12;      size:4; signed:0;
        field:int __probe_nargs;        offset:16;      size:4; signed:1;
        field:unsigned long dfd;        offset:20;      size:4; signed:0;
        field:unsigned long filename;   offset:24;      size:4; signed:0;
        field:unsigned long flags;      offset:28;      size:4; signed:0;
        field:unsigned long mode;       offset:32;      size:4; signed:0;


print fmt: "(%lx) dfd=%lx filename=%lx flags=%lx mode=%lx", REC->__probe_ip,
REC->dfd, REC->filename, REC->flags, REC->mode
```

You can see that the event has 4 arguments as in the expressions you specified.

```
  echo > /sys/kernel/debug/tracing/kprobe_events
```

This clears all probe points.

Or,

```
  echo -:myprobe >> kprobe_events
```

This clears probe points selectively.

Right after definition, each event is disabled by default. For tracing these events, you need to enable it.

```
  echo 1 > /sys/kernel/debug/tracing/events/kprobes/myprobe/enable
  echo 1 > /sys/kernel/debug/tracing/events/kprobes/myretprobe/enable
```

And you can see the traced information via /sys/kernel/debug/tracing/trace.

```
  cat /sys/kernel/debug/tracing/trace
# tracer: nop
#
#           TASK-PID    CPU#    TIMESTAMP  FUNCTION
#              | |       |          |         |
           <...>-1447  [001] 1038282.286875: myprobe: (do_sys_open+0x0/0xd6)
dfd=3 filename=7fffd1ec4440 flags=8000 mode=0
           <...>-1447  [001] 1038282.286878: myretprobe: (sys_openat+0xc/0xe <-
do_sys_open) $retval=fffffffffffffffe
           <...>-1447  [001] 1038282.286885: myprobe: (do_sys_open+0x0/0xd6)
dfd=ffffff9c filename=40413c flags=8000 mode=1b6
           <...>-1447  [001] 1038282.286915: myretprobe: (sys_open+0x1b/0x1d <-
do_sys_open) $retval=3
           <...>-1447  [001] 1038282.286969: myprobe: (do_sys_open+0x0/0xd6)
dfd=ffffff9c filename=4041c6 flags=98800 mode=10
           <...>-1447  [001] 1038282.286976: myretprobe: (sys_open+0x1b/0x1d <-
do_sys_open) $retval=3
```

Each line shows when the kernel hits an event, and <- SYMBOL means kernel returns from SYMBOL(e.g. "sys_open+0x1b/0x1d <- do_sys_open" means kernel returns from do_sys_open to sys_open+0x1b).