

Re: [PATCH][RFC] kprobes: Add user entry-handler in kretprobes

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-11/msg04725.html>

- *From:* Jim Keniston <jkenisto@xxxxxxxxxx>
 - *Date:* Wed, 14 Nov 2007 14:51:04 -0800
-

On Wed, 2007-11-14 at 19:00 +0530, Abhishek Sagar wrote:

First of all, some general comments. We seem to be trying to solve two problems here:

1. Prevent the asymmetry in entry- vs. return-handler calls that can develop when we temporarily run out of kretprobe_instances. E.g., if we have m kretprobe "misses", we may report n calls but only (n-m) returns.
2. Simplify the task of correlating data (e.g., timestamps) between function entry and function return.

Problem #1 wouldn't exist if we could solve problem #1a:

- 1a. Ensure that we never run out of kretprobe_instances (for some appropriate value of "never").

I've thought of various approaches to #1a -- e.g., allocate kretprobe_instances from GFP_ATOMIC memory when we're out of preallocated instances -- but I've never found time to pursue them. This might be a good time to brainstorm solutions to that problem.

Lacking a solution to #1a, I think Abhishek's approach provides a reasonable solution to problem #1.

I don't think it takes us very far toward solving #2, though. I agree with Srinivasa that it would be more helpful if we could store the data collected at function-entry time right in the kretprobe_instance. Kevin Stafford prototyped this "data pouch" idea a couple of years ago -- <http://sourceware.org/ml/systemtap/2005-q3/msg00593.html> -- but an analogous feature was implemented at a higher level in SystemTap. Either approach -- in kprobes or SystemTap -- would benefit from a fix to #1 (or #1a).

Review of Abhishek's patch:

I see no reason to save a copy of *regs and pass that to the entry handler. Passing the real regs pointer is good enough for other kprobe handlers. And if a handler on i386 uses ®s->esp as the value of the stack pointer (which is correct -- long story), it'll get the wrong value if its regs arg points at the copy.

Re: [PATCH][RFC] kprobes: Add user entry-handler in kretprobes

More comments below.

On Nov 14, 2007 3:53 PM, Srinivasa Ds <srinivasa@xxxxxxxxxx> wrote:

...

So entry_handler() which gets executed last doesn't guarantee that its return handler will be executed first(because it took a lot time to return).

Only if there are return instances pending belonging to different tasks.

I agree with Abhishek here.

...

Lets see how entry and return handlers can be matched up in three different scenarios:-

1. Multiple function entries from various tasks (the one you've just pointed out).
2. Multiple kretprobe registration on the same function.
3. Nested calls of kretprobe'd function.

In cases 1 and 3, the following information can be used to match corresponding entry and return handlers inside a user handler (if needed):

(ri->task, ri->ret_addr)
where ri is struct kretprobe_instance *

This tuple should uniquely identify a return address (right?).

But if it's a recursive function, there could be multiple instances in the same task with the same return address. The stack pointer would be different, FWIW.

In case 2, entry and return handlers are anyways called in the right order (taken care of by trampoline_handler() due to LIFO insertion in ri->hlist).

Re: [PATCH][RFC] kprobes: Add user entry-handler in kretprobes

Yes. And for case #2, ri->rp will be different for each kretprobe_instance anyway.

The fact that ri is passed to both handlers should allow any user handler to identify each of these cases and take appropriate synchronization action pertaining to its private data, if needed.

I don't think Abhishek has made his case here. See below.

(Hence I feel sol a) would be nice).

With an entry-handler, any module aiming to profile running time of a function (say) can simply do the following without being "return instance" conscious. Note however that I'm not trying to address just this scenario but trying to provide a general way to use entry-handlers in kretprobes:

```
static unsigned long flag = 0; /* use bit 0 as a global flag */
unsigned long long entry, exit;

int my_entry_handler(struct kretprobe_instance *ri, struct pt_regs *regs)
{
    if (!test_and_set_bit(0, &flag))
        /* this instance claims the first entry to kretprobe'd function */
        entry = sched_clock();
    /* do other stuff */
    return 0; /* right on! */
}

return 1; /* error: no return instance to be allocated for this
function entry */
}

/* will only be called iff flag == 1 */
int my_return_handler(struct kretprobe_instance *ri, struct pt_regs *regs)
{
    BUG_ON(!flag);
    exit = sched_clock();
    set_bit(0, &flag);
}
}
```

I think something like this should do the trick for you.

Re: [PATCH][RFC] kprobes: Add user entry-handler in kretprobes

Since flag is static, it seems to me that if there were instances of the probed function active concurrently in multiple tasks, only the first-called instance would be profiled.

Thanks
Srinivasa DS

--
Thanks & Regards
Abhishek Sagar

Jim Keniston

—
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>

Re: [PATCH][RFC] kprobes: Add user entry-handler in kretprobes