

Re: [PATCH -mm] markers: avoid call_rcu_sched if old is NULL

Re: [PATCH -mm] markers: avoid call_rcu_sched if old is NULL

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-07/msg03514.html>

- *From:* Mathieu Desnoyers <mathieu.desnoyers@xxxxxxxxxx>
 - *Date:* Wed, 9 Jul 2008 00:25:02 -0400
-

* Mathieu Desnoyers (mathieu.desnoyers@xxxxxxxxxx) wrote:

* Masami Hiramatsu (mhiramat@xxxxxxxxxx) wrote:

Introduce `marker_entry_free_old()` and check old pointer is NULL before setting `call_rcu_sched()`, because `marker_entry_remove/add_probe()` can return NULL.

Hi Masami,

I doubt this is a bug per se, because `kfree` accepts NULL pointers (and `kfree` is the only action done on the `oldptr` by `free_old_closure`).

This cleans up the code, so I think it's good to merge your patch, but I would definitely not classify this as a bugfix.

Acked-by: Mathieu Desnoyers <mathieu.desnoyers@xxxxxxxxxx>

Hrm, nope, finally there is a problem with your fix. Nack. Here is why :

Lets say we have two probes to connect on the marker, each with its own `private_data`.

If we pass from 1 connected probe (probe A) to 2 (Probes A and B), and then back to one (probe B), we want to make sure that readers (instrumentation sites) will see coherent probes (matching callback and private data). I remind you that 0 or 1 probes connected does not use an array with the markers. Only if 2 to N probes are connected do we use an array to store the callback and private data pairs. Therefore, special care must be taken of the callback and private data update in the 1 probe connected case (when there are 0 probes connected, the private data is not used and therefore we leave it as is at its old value).

However, because your cleanup actually removes the `rcu_barrier()` done

Re: [PATCH -mm] markers: avoid call_rcu_sched if old is NULL

Re: [PATCH -mm] markers: avoid call_rcu_sched if old is NULL

before the operation following the 2nd probe addition (the barrier is only done if there are rcu calls pending), we can end up doing :

Probes
A
{A, B}
B

within a single RCU period, which means that a reader can see :

A
B

Those are "single probes" which involves updates of two pointers : the callback and the private data. They must never be mixed within a single rcu period : valid transitions go from 0 to 1, 1 to 2, 2 N and from N to 2, 2 to 1, 1 to 0. The modification you are doing here adds transitions 1 to 1 (as my example show) which should never happen.

Final reminder : this "special case" of 0 and 1 probes connected is done to remove a pointer dereference and to minimize memory allocation in the "common case" where only 1 probe is connected to a marker. It's not used in the new tracepoint infrastructure because we have to iterate on the callbacks in the instrumentation sites. The markers can do this in a wrapper function, so we don't suffer from added instructions to every call sites. So your cleanup makes sense for the tracepoint infrastructure, but not for the markers.

Mathieu

Mathieu

Signed-off-by: Masami Hiramatsu <mhiramat@xxxxxxxxxxx>

kernel/marker.c | 29 ++++++-----
1 file changed, 14 insertions(+), 15 deletions(-)

Mathieu, I think this might be a bug. Tracepoint also has same bug...

Index: 2.6.26-rc8-mm1/kernel/marker.c

--- 2.6.26-rc8-mm1.orig/kernel/marker.c 2008-07-07
11:42:04.000000000 -0400
+++ 2.6.26-rc8-mm1/kernel/marker.c 2008-07-07 11:42:04.000000000
-0400
@@ -201,6 +201,17 @@ static void free_old_closure(struct rcu_
entry->rcu_pending = 0;

Re: [PATCH -mm] markers: avoid call_rcu_sched if old is NULL

Re: [PATCH -mm] markers: avoid call_rcu_sched if old is NULL

```
}

+static void marker_entry_free_old(struct marker_entry *entry, void *old)
+{
+ if (!old)
+ return;
+ entry->oldptr = old;
+ entry->rcu_pending = 1;
+ /* write rcu_pending before calling the RCU callback */
+ smp_wmb();
+ call_rcu_sched(&entry->rcu, free_old_closure);
+}
+
static void debug_print_probes(struct marker_entry *entry)
{
int i;
@@ -666,11 +677,7 @@ int marker_probe_register(const char *name,
mutex_lock(&markers_mutex);
entry = get_marker(name);
WARN_ON(!entry);
- entry->oldptr = old;
- entry->rcu_pending = 1;
- /* write rcu_pending before calling the RCU callback */
- smp_wmb();
- call_rcu_sched(&entry->rcu, free_old_closure);
+ marker_entry_free_old(entry, old);
end:
mutex_unlock(&markers_mutex);
return ret;
@@ -709,11 +716,7 @@ int marker_probe_unregister(const char *name)
entry = get_marker(name);
if (!entry)
goto end;
- entry->oldptr = old;
- entry->rcu_pending = 1;
- /* write rcu_pending before calling the RCU callback */
- smp_wmb();
- call_rcu_sched(&entry->rcu, free_old_closure);
+ marker_entry_free_old(entry, old);
remove_marker(name); /* Ignore busy error message */
ret = 0;
end:
@@ -787,11 +790,7 @@ int marker_probe_unregister_private_data(const char *name,
mutex_lock(&markers_mutex);
entry = get_marker_from_private_data(probe, probe_private);
WARN_ON(!entry);
- entry->oldptr = old;
- entry->rcu_pending = 1;
- /* write rcu_pending before calling the RCU callback */
- smp_wmb();
- call_rcu_sched(&entry->rcu, free_old_closure);
```

Re: [PATCH -mm] markers: avoid call_rcu_sched if old is NULL

Re: [PATCH -mm] markers: avoid call_rcu_sched if old is NULL

```
+ marker_entry_free_old(entry, old);  
remove_marker(entry->name); /* Ignore busy error message */  
end:  
mutex_unlock(&markers_mutex);  
--
```

Masami Hiramatsu

Software Engineer
Hitachi Computer Products (America) Inc.
Software Solutions Division

e-mail: mhiramat@xxxxxxxxxxx

--
Mathieu Desnoyers

OpenPGP key fingerprint: 8CD5 52C3 8E3C 4140 715F BA06 3F25 A8FE 3BAE 9A68

--
Mathieu Desnoyers

OpenPGP key fingerprint: 8CD5 52C3 8E3C 4140 715F BA06 3F25 A8FE 3BAE 9A68

--
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>