Subject: Re: [PATCH v4 1/3] make jump_labels wait while updates are in place
Posted by KAMEZAWA Hiroyuki on Fri, 27 Apr 2012 01:05:02 GMT
View Forum Message <> Reply to Message

(2012/04/27 9:43), Steven Rostedt wrote:

```
> On Thu, Apr 26, 2012 at 07:51:05PM -0300, Glauber Costa wrote:
>> In mem cgroup, we need to guarantee that two concurrent updates
>> of the jump_label interface wait for each other. IOW, we can't have
>> other updates returning while the first one is still patching the
>> kernel around, otherwise we'll race.
>
> But it shouldn't. The code as is should prevent that.
>
>>
>> I believe this is something that can fit well in the static branch
>> API, without noticeable disadvantages:
>>
>> * in the common case, it will be a quite simple lock/unlock operation
>> * Every context that calls static_branch_slow* already expects to be
>>   in sleeping context because it will mutex_lock the unlikely case.
>> * static_key_slow_inc is not expected to be called in any fast path,
>>   otherwise it would be expected to have quite a different name. Therefore
>>   the mutex + atomic combination instead of just an atomic should not kill
>>   us.
>>
>> Signed-off-by: Glauber Costa <glommer@parallels.com>
>> CC: Tejun Heo <tj@kernel.org>
>> CC: Li Zefan <lizefan@huawei.com>
>> CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
>> CC: Johannes Weiner <hannes@cmpxchg.org>
>> CC: Michal Hocko <mhocko@suse.cz>
>> CC: Ingo Molnar <mingo@elte.hu>
>> CC: Jason Baron <jbaron@redhat.com>
>> ---
>>  kernel/jump_label.c |   21 +++++++++++----------
>>  1 files changed, 11 insertions(+), 10 deletions(-)
>>
>> diff --git a/kernel/jump_label.c b/kernel/jump_label.c
>> index 4304919..5d09cb4 100644
>> --- a/kernel/jump_label.c
>> +++ b/kernel/jump_label.c
>> @@ -57,17 +57,16 @@ static void jump_label_update(struct static_key *key, int enable);
>>
>>  void static_key_slow_inc(struct static_key *key)
>>  {
>> + jump_label_lock();
>>   if (atomic_inc_not_zero(&key->enabled))
```

>> - return;
>
> If key->enabled is not zero, there's nothing to be done. As the jump
> label has already been enabled. Note, the key->enabled doesn't get set
> until after the jump label is updated. Thus, if two tasks were to come
> in, they both would be locked on the jump_label_lock().
>

Ah, sorry, I misunderstood somthing. I'm sorry, Glauber.

-Kame