Subject: Re: [PATCH v5 2/2] decrement static keys on real destroy time
Posted by Glauber Costa on Thu, 17 May 2012 10:22:05 GMT
View Forum Message <> Reply to Message

On 05/17/2012 02:18 PM, KAMEZAWA Hiroyuki wrote:
> (2012/05/17 18:52), Glauber Costa wrote:
>
>> On 05/17/2012 09:37 AM, Andrew Morton wrote:
>>>>> If that happens, locking in static_key_slow_inc will prevent any damage.
>>>>> My previous version had explicit code to prevent that, but we were
>>>>> pointed out that this is already part of the static_key expectations, so
>>>>> that was dropped.
>>> This makes no sense. If two threads run that code concurrently,
>>> key->enabled gets incremented twice. Nobody anywhere has a record that
>>> this happened so it cannot be undone. key->enabled is now in an
>>> unknown state.
>>
>> Kame, Tejun,
>>
>> Andrew is right. It seems we will need that mutex after all. Just this
>> is not a race, and neither something that should belong in the
>> static_branch interface.
>>
>
>
> Hmm....how about having
>
> res_counter_xchg_limit(res,&old_limit, new_limit);
>
> if (!cg_proto->updated&& old_limit == RESOURCE_MAX)
> ....update labels...
>
> Then, no mutex overhead maybe and activated will be updated only once.
> Ah, but please fix in a way you like. Above is an example.

I think a mutex is a lot cleaner than adding a new function to the
res_counter interface.

We could do a counter, and then later decrement the key until the
counter reaches zero, but between those two, I still think a mutex here
is preferable.

Only that, instead of coming up with a mutex of ours, we could export
and reuse set_limit_mutex from memcontrol.c

> Thanks,
> -Kame

> (*) I'm sorry I won't be able to read e-mails, tomorrow.
>
Ok Kame. I am not in a terrible hurry to fix this, it doesn't seem to be hurting any real workload.