Subject: Re: [PATCH v2 19/29] skip memcg kmem allocations in specified code regions
Posted by KAMEZAWA Hiroyuki on Wed, 16 May 2012 07:55:49 GMT
View Forum Message <> Reply to Message

(2012/05/16 15:19), Glauber Costa wrote:

```
> On 05/15/2012 06:46 AM, KAMEZAWA Hiroyuki wrote:
>> (2012/05/12 2:44), Glauber Costa wrote:
>>
>>> This patch creates a mechanism that skip memcg allocations during
>>> certain pieces of our core code. It basically works in the same way
>>> as preempt_disable()/preempt_enable(): By marking a region under
>>> which all allocations will be accounted to the root memcg.
>>>
>>> We need this to prevent races in early cache creation, when we
>>> allocate data using caches that are not necessarily created already.
>>>
>>> Signed-off-by: Glauber Costa<glommer@parallels.com>
>>> CC: Christoph Lameter<cl@linux.com>
>>> CC: Pekka Enberg<penberg@cs.helsinki.fi>
>>> CC: Michal Hocko<mhocko@suse.cz>
>>> CC: Kamezawa Hiroyuki<kamezawa.hiroyu@jp.fujitsu.com>
>>> CC: Johannes Weiner<hannes@cmpxchg.org>
>>> CC: Suleiman Souhlal<suleiman@google.com>
>>
>>
>> The concept seems okay to me but...
>>
>>> ---
>>>   include/linux/sched.h |    1 +
>>>   mm/memcontrol.c       |   25 +++++++++++++++++++++++++
>>>   2 files changed, 26 insertions(+), 0 deletions(-)
>>>
>>> diff --git a/include/linux/sched.h b/include/linux/sched.h
>>> index 81a173c..0501114 100644
>>> --- a/include/linux/sched.h
>>> +++ b/include/linux/sched.h
>>> @@ -1613,6 +1613,7 @@ struct task_struct {
>>>     unsigned long nr_pages; /* uncharged usage */
>>>     unsigned long memsw_nr_pages; /* uncharged mem+swap usage */
>>>    } memcg_batch;
>>> + atomic_t memcg_kmem_skip_account;
>>
>>
>> If only 'current' thread touch this, you don't need to make this atomic counter.
>> you can use 'long'.
>>
```

> You're absolutely right, Kame, thanks.
> I first used atomic_t because I had it tested against current->mm->owner.
>
> Do you, btw, agree to use current instead of owner here?
> You can find the rationale in earlier mails between me and Suleiman.

I agree to use current. This information depends on the context of callers.

Thanks,
-Kame