Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Glauber Costa on Mon, 25 Jun 2012 12:55:31 GMT
View Forum Message <> Reply to Message

On 06/25/2012 04:49 PM, Michal Hocko wrote:
> On Mon 25-06-12 16:11:01, Glauber Costa wrote:
>> On 06/25/2012 04:08 PM, Michal Hocko wrote:
>>> On Mon 25-06-12 13:21:01, Glauber Costa wrote:
> [...]
>>>> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
>>>> index ac35bcc..cccebbc 100644
>>>> --- a/mm/memcontrol.c
>>>> +++ b/mm/memcontrol.c
>>>> @@ -3779,6 +3779,10 @@ static int mem_cgroup_hierarchy_write(struct cgroup *cont,
struct cftype *cft,
>>>> parent_memcg = mem_cgroup_from_cont(parent);
>>>>
>>>> cgroup_lock();
>>>> +
>>>> + if (memcg->use_hierarchy == val)
>>>> + goto out;
>>>> +
>>>
>>> Why do you need cgroup_lock to check the value? Even if we have 2
>>> CPUs racing (one trying to set to 0 other to 1 with use_hierarchy==0)
>>> then the "set to 0" operation might fail depending on who hits the
>>> cgroup_lock first anyway.
>>>
>>> So while this is correct I think there is not much point to take the global
>>> cgroup lock in this case.
>>>
>> Well, no.
>>
>> All operations will succeed, unless the cgroup breeds new children.
>> That's the operation we're racing against.
>
> I am not sure I understand. The changelog says that you want to handle
> a situation where you are copying a hierarchy along with their
> attributes and you don't want to fail when setting sane values.
>
> If we race with a new child creation then the success always depends on
> the lock ordering but once the value is set then it is final so the test
> will work even outside of the lock. Or am I still missing something?
>
> Just to make it clear the lock is necessary in the function I just do
> not see why it should be held while we are trying to handle no-change
> case.
>

I think you are right in this specific case. But do you think it is necessary to submit a version of it that tests outside the lock?

We don't gain too much with that anyway.