
Subject: Re: [PATCH v3 12/13] execute the whole memcg freeing in rcu callback
Posted by [Glauber Costa](#) on Thu, 04 Oct 2012 10:53:13 GMT
[View Forum Message](#) <> [Reply to Message](#)

On 10/01/2012 05:27 PM, Michal Hocko wrote:

> On Tue 18-09-12 18:04:09, Glauber Costa wrote:

>> A lot of the initialization we do in mem_cgroup_create() is done with softirqs
>> enabled. This include grabbing a css id, which holds &ss->id_lock->rlock, and
>> the per-zone trees, which holds rtpz->lock->rlock. All of those signal to the
>> lockdep mechanism that those locks can be used in SOFTIRQ-ON-W context. This
>> means that the freeing of memcg structure must happen in a compatible context,
>> otherwise we'll get a deadlock.

>

> Maybe I am missing something obvious but why cannot we simply disable
> (soft)irqs in mem_cgroup_create rather than make the free path much more
> complicated. It really feels strange to defer everything (e.g. soft
> reclaim tree cleanup which should be a no-op at the time because there
> shouldn't be any user pages in the group).

>

Ok.

I was just able to come back to this today - I was mostly working on the
slab feedback over the past few days. I will answer yours and Tejun's
concerns at once:

Here is the situation: the backtrace I get is this one:

```
[ 124.956725] =====  
[ 124.957217] [ INFO: inconsistent lock state ]  
[ 124.957217] 3.5.0+ #99 Not tainted  
[ 124.957217] -----  
[ 124.957217] inconsistent {SOFTIRQ-ON-W} -> {IN-SOFTIRQ-W} usage.  
[ 124.957217] ksoftirqd/0/3 [HC0[0]:SC1[1]:HE1:SE0] takes:  
[ 124.957217] (&(&ss->id_lock)->rlock){+?.?...}, at:  
[<ffffffff810aa7b2>] spin_lock+0x9/0xb  
[ 124.957217] {SOFTIRQ-ON-W} state was registered at:  
[ 124.957217] [<ffffffff810996ed>] __lock_acquire+0x31f/0xd68  
[ 124.957217] [<ffffffff8109a660>] lock_acquire+0x108/0x15c  
[ 124.957217] [<ffffffff81534ec4>] _raw_spin_lock+0x40/0x4f  
[ 124.957217] [<ffffffff810aa7b2>] spin_lock+0x9/0xb  
[ 124.957217] [<ffffffff810ad00e>] get_new_cssid+0x69/0xf3  
[ 124.957217] [<ffffffff810ad0da>] cgroup_init_idr+0x42/0x60  
[ 124.957217] [<ffffffff81b20e04>] cgroup_init+0x50/0x100  
[ 124.957217] [<ffffffff81b05b9b>] start_kernel+0x3b9/0x3ee  
[ 124.957217] [<ffffffff81b052d6>] x86_64_start_reservations+0xb1/0xb5  
[ 124.957217] [<ffffffff81b053d8>] x86_64_start_kernel+0xfe/0x10b
```

So what we learn from it, is: we are acquiring a specific lock (the css id one) from softirq context. It was previously taken in a softirq-enabled context, that seems to be coming directly from `get_new_cssid`.

Tejun correctly pointed out that we should never acquire that lock from a softirq context, in which he is right.

But the situation changes slightly with kmem. Now, the following excerpt of a backtrace is possible:

```
[ 48.602775] [<ffffffff81103095>] free_accounted_pages+0x47/0x4c
[ 48.602775] [<ffffffff81047f90>] free_task+0x31/0x5c
[ 48.602775] [<ffffffff8104807d>] __put_task_struct+0xc2/0xdb
[ 48.602775] [<ffffffff8104dfc7>] put_task_struct+0x1e/0x22
[ 48.602775] [<ffffffff8104e144>] delayed_put_task_struct+0x7a/0x98
[ 48.602775] [<ffffffff810cf0e5>] __rcu_process_callbacks+0x269/0x3df
[ 48.602775] [<ffffffff810cf28c>] rcu_process_callbacks+0x31/0x5b
[ 48.602775] [<ffffffff8105266d>] __do_softirq+0x122/0x277
```

So as you can see, `free_accounted_pages` (that will trigger a `memcg_put()` -> `mem_cgroup_free()`) can now be called from softirq context, which is, an rcu callback (and I just realized I wrote the exact opposite in the subj line: man, I really suck at that!!)
As a matter of fact, we could not move to our rcu callback as well:

we need to move it to a worker thread with the rest.

We already have a worker thread: the reason we have it is not `static_branches`: The reason is `vfree()`, that will `BUG_ON(in_interrupt())` and could not be called from rcu callback as well. We moved static branches in there as well for a similar problem, but haven't introduced it.

Could we move just part of it to the worker thread? Absolutely yes. Moving just `free_css_id()` is enough to make it work. But since it is not the first context related problem we had, I thought: "to hell with that, let's move everything and be safe".

I am fine moving `free_css_id()` only if you would prefer.

Can we disable softirqs when we initialize `css_id`? Maybe. My machine seems to boot fine and survive the simple workload that would trigger that bug if I use `irqsave spinlocks` instead of normal spinlocks. But this has to be done from cgroup core: We have no control over css creation in `memcg`.

How would you guys like me to handle this ?
