Subject: Re: [PATCH v5 2/2] decrement static keys on real destroy time
Posted by KAMEZAWA Hiroyuki on Thu, 17 May 2012 00:07:11 GMT
View Forum Message <> Reply to Message

(2012/05/17 6:13), Andrew Morton wrote:

> On Fri, 11 May 2012 17:11:17 -0300
> Glauber Costa <glommer@parallels.com> wrote:
>
>> We call the destroy function when a cgroup starts to be removed,
>> such as by a rmdir event.
>>
>> However, because of our reference counters, some objects are still
>> inflight. Right now, we are decrementing the static_keys at destroy()
>> time, meaning that if we get rid of the last static_key reference,
>> some objects will still have charges, but the code to properly
>> uncharge them won't be run.
>>
>> This becomes a problem specially if it is ever enabled again, because
>> now new charges will be added to the staled charges making keeping
>> it pretty much impossible.
>>
>> We just need to be careful with the static branch activation:
>> since there is no particular preferred order of their activation,
>> we need to make sure that we only start using it after all
>> call sites are active. This is achieved by having a per-memcg
>> flag that is only updated after static_key_slow_inc() returns.
>> At this time, we are sure all sites are active.
>>
>> This is made per-memcg, not global, for a reason:
>> it also has the effect of making socket accounting more
>> consistent. The first memcg to be limited will trigger static_key()
>> activation, therefore, accounting. But all the others will then be
>> accounted no matter what. After this patch, only limited memcgs
>> will have its sockets accounted.
>
> So I'm scratching my head over what the actual bug is, and how
> important it is.  AFAICT it will cause charging stats to exhibit some
> inaccuracy when memcg's are being torn down?
>
> I don't know how serious this in in the real world and so can't decide
> which kernel version(s) we should fix.
>
> When fixing bugs, please always fully describe the bug's end-user
> impact, so that I and others can make these sorts of decisions.
>

Ah, this was a bug report from me. tcp accounting can be easily broken.
Costa, could you include this ?
==

tcp memcontrol uses static_branch to optimize limit=RESOURCE_MAX case.
If all cgroup's limit=RESOUCE_MAX, resource usage is not accounted.
But it's buggy now.

For example, do following
# while sleep 1;do
  echo 9223372036854775807 > /cgroup/memory/A/memory.kmem.tcp.limit_in_bytes;
  echo 300M > /cgroup/memory/A/memory.kmem.tcp.limit_in_bytes;
  done
and run network application under A. tcp's usage is sometimes accounted
and sometimes not accounted because of frequent changes of static_branch.
Then,  you can see broken tcp.usage_in_bytes.
WARN_ON() is printed because res_counter->usage goes below 0.
==
kernel: ------------[ cut here ]----------
kernel: WARNING: at kernel/res_counter.c:96 res_counter_uncharge_locked+0x37/0x40()
 <snip>
kernel: Pid: 17753, comm: bash Tainted: G  W    3.3.0+ #99
kernel: Call Trace:
kernel: <IRQ>  [<ffffffff8104cc9f>] warn_slowpath_common+0x7f/0xc0
kernel: [<ffffffff810d7e88>] ? rb_reserve__next_event+0x68/0x470
kernel: [<ffffffff8104ccfa>] warn_slowpath_null+0x1a/0x20
kernel: [<ffffffff810b4e37>] res_counter_uncharge_locked+0x37/0x40
 ...
==