Subject: Re: [PATCH v5 0/8] per-cgroup tcp buffer pressure settings
Posted by Glauber Costa on Fri, 07 Oct 2011 08:20:04 GMT
View Forum Message <> Reply to Message

On 10/07/2011 12:05 PM, KAMEZAWA Hiroyuki wrote:
>
>
> Sorry for lazy answer.
Hi Kame,

Now matter how hard you try, you'll never be as lazy as I am. So that's
okay.

>
> On Wed, 5 Oct 2011 11:25:50 +0400
> Glauber Costa<glommer@parallels.com> wrote:
>
>> On 10/05/2011 04:29 AM, KAMEZAWA Hiroyuki wrote:
>>> On Tue, 4 Oct 2011 16:17:52 +0400
>>> Glauber Costa<glommer@parallels.com>   wrote:
>>>
>
>>> At this stage, my concern is view of interfaces and documenation, and future plans.
>>
>> Okay. I will try to address them as well as I can.
>>
>>> * memory.independent_kmem_limit
>>>   If 1, kmem_limit_in_bytes/kmem_usage_in_bytes works.
>>>   If 0, kmem_limit_in_bytes/kmem_usage_in_bytes doesn't work and all kmem
>>>   usages are controlled under memory.limit_in_bytes.
>>
>> Correct. For the questions below, I won't even look at the code not to
>> get misguided. Let's settle on the desired behavior, and everything that
>> deviates from it, is a bug.
>>
>>> Question:
>>>   - What happens when parent/chidlren cgroup has different indepedent_kmem_limit ?
>> I think it should be forbidden. It was raised by Kirill before, and
>> IIRC, he specifically requested it to be. (Okay: Saying it now, makes me
>> realizes that the child can have set it to 1 while parent was 1. But
>> then parent sets it to 0... I don't think I am handling this case).
>>
>
> ok, please put it into TODO list ;)

Done.

>

>
>>> In future plan, kmem.usage_in_bytes should includes tcp.kmem_usage_in_bytes.
>>> And kmem.limit_in_bytes should be the limiation of sum of all kmem.xxxx.limit_in_bytes.
>>
>> I am not sure there will be others xxx.limit_in_bytes. (see below)
>>
>
> ok.
>
>
>>>
>>> Question:
>>> - Why this integration is difficult ?
>> It is not that it is difficult.
>> What happens is that there are two things taking place here:
>> One of them is allocation.
>> The other, is tcp-specific pressure thresholds. Bear with me with the
>> following example code: (from sk_stream_alloc_skb, net/ipv4/tcp.c)
>>
>> 1:   skb = alloc_skb_fclone(size + sk->sk_prot->max_header, gfp);
>>   if (skb) {
>> 3:   if (sk_wmem_schedule(sk, skb->truesize)) {
>>   /*
>>   * Make sure that we have exactly size bytes
>>   * available to the caller, no more, no less.
>>   */
>>   skb_reserve(skb, skb_tailroom(skb) - size);
>>   return skb;
>>   }
>>   __kfree_skb(skb);
>>   } else {
>>   sk->sk_prot->enter_memory_pressure(sk);
>>   sk_stream_moderate_sndbuf(sk);
>>   }
>>
>> In line 1, an allocation takes place. This allocs memory from the skbuff
>> slab cache.
>> But then, pressure thresholds are applied in 3. If it fails, we drop the
>> memory buffer even if the allocation succeeded.
>>
>
> Sure.
>
>
>> So this patchset, as I've stated already, cares about pressure
>> conditions only. It is enough to guarantee that no more memory will be
>> pinned that we specified, because we'll free the allocation in case
>> pressure is reached.

>>
>> There is work in progress from guys at google (and I have my very own
>> PoCs as well), to include all slab allocations in kmem.usage_in_bytes.
>>
>
> ok.
>
>
>> So what I really mean here with "will integrate later", is that I think
>> that we'd be better off tracking the allocations themselves at the slab
>> level.
>>
>>>> Can't tcp-limit-code borrows some amount of charges in batch from kmem_limit
>>>> and use it ?
>> Sorry, I don't know what exactly do you mean. Can you clarify?
>>
> Now, tcp-usage is independent from kmem-usage.
>
> My idea is
>
> 1. when you account tcp usage, charge kmem, too.

Absolutely.
> Now, your work is
> a) tcp use new xxxx bytes.
> b) account it to tcp.uage and check tcp limit
>
> To ingegrate kmem,
> a) tcp use new xxxx bytes.
> b) account it to tcp.usage and check tcp limit
> c) account it to kmem.usage
>
> ? 2 counters may be slow ?

Well, the way I see it, 1 counter is slow already =)
I honestly think we need some optimizations here. But
that is a side issue.

To begin with: The new patchset that I intend to spin
today or Monday, depending on my progress, uses res_counters,
as you and Kirill requested.

So what makes res_counters slow IMHO, is two things:

1) interrupts are always disabled.
2) All is done under a lock.

Now, we are starting to have resources that are billed to multiple

counters. One simple way to work around it, is to have child counters
that has to be accounted for as well everytime a resource is counted.

Like this:

1) tcp has kmem as child. When we bill to tcp, we bill to kmem as well.
   For protocols that do memory pressure, we then don't bill kmem from
   the slab.
2) When kmem_independent_account is set to 0, kmem has mem as child.

>
>
>>> - Don't you need a stat file to indicate "tcp memory pressure works!" ?
>>>   It can be obtained already ?
>>
>> Not 100 % clear as well. We can query the amount of buffer used, and the
>> amount of buffer allowed. What else do we need?
>>
>
> IIUC, we can see the fact tcp.usage is near to tcp.limit but never can see it
> got memory pressure and how many numbers of failure happens.
> I'm sorry if I don't read codes correctly.

IIUC, With res_counters being used, we get at least failcnt for free, right?
