Subject: Re: [PATCH v5 0/8] per-cgroup tcp buffer pressure settings
Posted by Glauber Costa on Wed, 05 Oct 2011 07:25:50 GMT
View Forum Message <> Reply to Message

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:
>
>> [[v3: merge Kirill's suggestions, + a destroy-related bugfix]]
>> [[v4: Fix a bug with non-mounted cgroups + disallow task movement]]
>> [[v5: Compile bug with modular ipv6 + tcp files in bytes]]
>>
>> Kame, Kirill,
>>
>> I am submitting this again merging most of your comments. I've decided to
>> leave some of them out:
>> * I am not using res_counters for allocated_memory. Besides being more
>> expensive than what we need, to make it work in a nice way, we'd have
>> to change the !cgroup code, including other protocols than tcp. Also,
>>
>> * I am not using failcnt and max_usage_in_bytes for it. I believe the value
>> of those lies more in the allocation than in the pressure control. Besides,
>> fail conditions lie mostly outside of the memory cgroup's control. (Actually,
>> a soft_limit makes a lot of sense, and I do plan to introduce it in a follow
>> up series)
>>
>> If you agree with the above, and there are any other pressing issues, let me
>> know and I will address them ASAP. Otherwise, let's discuss it. I'm always open.
>>
>
> I'm not familar with reuqirements of users. So, I appreciate your choices.
> What I adivse you here is taking a deep breath. Making new version every day
> is not good for reviewing process ;)
> (It's now -rc8 and merge will not be so quick, anyway.)

Kame,

Absolutely. I only did it this time because the difference between them
were really, really small.

> 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.

> Let me give a try explanation by myself. (Correct me ;)
> I added some questions but I'm sorry you've already answered.
>
> New interfaces are 5 files. All files exists only for non-root memory cgroup.

>
> 1. memory.independent_kmem_limit
> 2. memory.kmem.usage_in_bytes
> 3. memory.kmem.limit_in_bytes
> 4. memory.kmem.tcp.limit_in_bytes
> 5. memory.kmem.tcp.usage_in_bytes

Correct so far. Note that the tcp memory pressure parameters right now
consists of 3 numbers, one of them being a soft limit. I plan to add the
soft limit file in a follow up patch, to avoid adding more and more
stuff for us to review here. (Unless of course you want to see it now)

> * 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).

> - What happens at creating a new cgroup with use_hierarchy==1.
>
> * memory.kmem_limit_in_bytes/memory.kmem.tcp.limit_in_bytes
>
> Both files works independently for _Now_. And memory.kmem_usage_in_bytes and
> memory.kmem_tcp.usage_in_bytes has no relationships.

Correct.

> 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)

>
> 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.

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.

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?

> - 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?