Subject: Re: Re: [PATCH v5 00/10] per-cgroup tcp memory pressure
Posted by Glauber Costa on Wed, 23 Nov 2011 10:25:17 GMT
View Forum Message <> Reply to Message

On 11/22/2011 12:07 AM, KAMEZAWA Hiroyuki wrote:
> On Fri, 18 Nov 2011 17:39:03 -0200
> Glauber Costa<glommer@parallels.com> wrote:
>
>> On 11/17/2011 07:35 PM, David Miller wrote:
>>> TCP specific stuff in mm/memcontrol.c, at best that's not nice at all.
>>
>> How crucial is that? Thing is that as far as I am concerned, all the
>> memcg people really want the inner layout of struct mem_cgroup to be
>> private to memcontrol.c
>
> This is just because memcg is just related to memory management and I don't
> want it be wide spreaded, 'struct mem_cgroup' has been changed often.
>
> But I don't like to have TCP code in memcgroup.c.
>
> New idea is welcome.
I don't like it either, but it seemed like an acceptable idea when I
first wrote it, compared to the options.
But I'm happy it was strongly pointed out, I gave this some extra
thought, and managed to come up with a more elegant solution for this.
Unfortunately, right now I still have some very small tcp glue code I am
trying to get rid of - maybe the suggestion you give below can be used

>> This means that at some point, we need to have
>> at least a wrapper in memcontrol.c that is able to calculate the offset
>> of the tcp structure, and since most functions are actually quite
>> simple, that would just make us do more function calls.
>>
>> Well, an alternative to that would be to use a void pointer in the newly
>> added struct cg_proto to an already parsed memcg-related field
>> (in this case tcp_memcontrol), that would be passed to the functions
>> instead of the whole memcg structure. Do you think this would be
>> preferable ?
>>
In the patch I have now, I changed cg_proto's role a bit. It now have
pointers to the variables directly as well, and a parent pointer. It
allows much of the code to be simplified, and the remainder to live
outside memcontrol.c without major problems.

But I still need a tcp control structure and a method to calculate its
address from a mem_cgroup structure.

> like this ?

>
> struct mem_cgroup_sub_controls {
> struct mem_cgroup *mem;
> union {
>   struct tcp_mem_control tcp;
> } data;
> };
> /* for loosely coupled controls for memcg */
> struct memcg_sub_controls_function
> {
> struct memcg_sub_controls (*create)(struct mem_cgroup *);
> struct memcg_sub_controls (*destroy)(struct mem_cgroup *);
> }
>
> int register_memcg_sub_controls(char *name,
>   struct memcg_sub_controls_function *abis);
>
>
> struct mem_cgroup {
> .....
> .....
> /* Root memcg will have no sub_controls! */
> struct memcg_sub_controls *sub_controls[NR_MEMCG_SUB_CONTROLS];
> }
>
>
> Maybe some functions should be exported.
>
> Thanks,
> -Kame

This has the disadvantage that we now have to chase one more pointer
(for sub_controls) instead of just accessing tcp_memcontrol directly as in

struct mem_cgroup {
 ...
 struct tcp_memcontrol tcp;
};

I see how it would be nice to get rid of all tcp references, but I don't
think the above is worse that bundling struct tcp_memcontrol in an union.
We also now have to allocate mem_cgroup_subcontrols separately, creating
another opportunity of failure (that's not the end of the world, but
allocating it all inside memcg is still preferred, IMHO)

This could work if it was more generic than that, with a data pointer,
instead of an union. But then we now have 3 pointers to chase.
My worry here is that we'll end up with a performance a lot worse than

the raw network. Well, of course we'll be worse, but in the end of the day, we also want to be as fast as we can.

I wonder how bad it is to have just a tiny glue that calculates the address of the tcp structure in memcontrol.c, and then every other user is passed that structure?