Subject: Re: [PATCH v2 04/11] kmem accounting basic infrastructure
Posted by Glauber Costa on Wed, 15 Aug 2012 12:53:40 GMT
View Forum Message <> Reply to Message

On 08/15/2012 04:39 PM, Michal Hocko wrote:
> On Wed 15-08-12 13:33:55, Glauber Costa wrote:
> [...]
>>> This can
>>> be quite confusing. I am still not sure whether we should mix the two
>>> things together. If somebody wants to limit the kernel memory he has to
>>> touch the other limit anyway. Do you have a strong reason to mix the
>>> user and kernel counters?
>>
>> This is funny, because the first opposition I found to this work was
>> "Why would anyone want to limit it separately?" =p
>>
>> It seems that a quite common use case is to have a container with a
>> unified view of "memory" that it can use the way he likes, be it with
>> kernel memory, or user memory. I believe those people would be happy to
>> just silently account kernel memory to user memory, or at the most have
>> a switch to enable it.
>>
>> What gets clear from this back and forth, is that there are people
>> interested in both use cases.
>
> I am still not 100% sure myself. It is just clear that the reclaim would
> need some work in order to do accounting like this.
>

Note: Besides what I've already said, right *now* in this series we are
accounting just stack. So reclaimable vs not-reclaimable doesn't even
get to play. It is used while the tasks are running, it gets freed after
the tasks exited.

I do agree we need to look to the whole picture, and reclaiming will be
hard to get right.

This is actually why we're addressing them separately: because they are
a hard problem on their own, and the current status of accounting
already solve real life problems for many, though not for all.

>>> My impression was that kernel allocation should simply fail while user
>>> allocations might reclaim as well. Why should we reclaim just because of
>>> the kernel allocation (which is unreclaimable from hard limit reclaim
>>> point of view)?
>>
>> That is not what the kernel does, in general. We assume that if he wants
>> that memory and we can serve it, we should. Also, not all kernel memory

>> is unreclaimable. We can shrink the slabs, for instance. Ying Han
>> claims she has patches for that already...
>
> Are those patches somewhere around?
>

Ying Han ?

> [...]
>>> This doesn't check for the hierachy so kmem_accounted might not be in
>>> sync with it's parents. mem_cgroup_create (below) needs to copy
>>> kmem_accounted down from the parent and the above needs to check if this
>>> is a similar dance like mem_cgroup_oom_control_write.
>>>
>>
>> I don't see why we have to.
>>
>> I believe in a A/B/C hierarchy, C should be perfectly able to set a
>> different limit than its parents. Note that this is not a boolean.
>
> Ohh, I wasn't clear enough. I am not against setting the _limit_ I just
> meant that the kmem_accounted should be consistent within the hierarchy.
>

If a parent of yours is accounted, you get accounted as well. This is
not the state in this patch, but gets added later. Isn't this enough ?