Subject: Re: [PATCH v2 09/11] memcg: propagate kmem limiting information to children

Posted by Glauber Costa on Fri, 24 Aug 2012 05:23:58 GMT

View Forum Message <> Reply to Message

On 08/24/2012 09:06 AM, Greg Thelen wrote:
> On Thu, Aug 23 2012, Glauber Costa wrote:
>
>> On 08/23/2012 03:23 AM, Greg Thelen wrote:
>>> On Wed, Aug 22 2012, Glauber Costa wrote:
>>>
>>>>>>>>
>>>>>>>> I am fine with either, I just need a clear sign from you guys so I don't
>>>>>>>> keep deimplementing and reimplementing this forever.
>>>>>>
>>>>>> I would be for make it simple now and go with additional features later
>>>>>> when there is a demand for them. Maybe we will have runtimg switch for
>>>>>> user memory accounting as well one day.
>>>>>>
>>>>>> But let's see what others think?
>>>>>
>>>>> In my use case memcg will either be disable or (enabled and kmem
>>>>> limiting enabled).
>>>>>
>>>>> I'm not sure I follow the discussion about history. Are we saying that
>>>>> once a kmem limit is set then kmem will be accounted/charged to memcg.
>>>>> Is this discussion about the static branches/etc that are autotuned the
>>>>> first time is enabled?
>>>>
>>>> No, the question is about when you unlimit a former kmem-limited memcg.
>>>>
>>>>> The first time its set there parts of the system
>>>>> will be adjusted in such a way that may impose a performance overhead
>>>>> (static branches, etc). Thereafter the performance cannot be regained
>>>>> without a reboot. This makes sense to me. Are we saying that
>>>>> kmem.limit_in_bytes will have three states?
>>>>
>>>> It is not about performance, about interface.
>>>>
>>>> Michal says that once a particular memcg was kmem-limited, it will keep
>>>> accounting pages, even if you make it unlimited. The limits won't be
>>>> enforced, for sure - there is no limit, but pages will still be accounted.
>>>>
>>>> This simplifies the code galore, but I worry about the interface: A
>>>> person looking at the current status of the files only, without
>>>> knowledge of past history, can't tell if allocations will be tracked or not.
>>>
>>> In the current patch set we've conflating enabling kmem accounting with

>>> the kmem limit value (RESOURCE_MAX=disabled, all_other_values=enabled).
>>>
>>> I see no problem with simpling the kernel code with the requirement that
>>> once a particular memcg enables kmem accounting that it cannot be
>>> disabled for that memcg.
>>>
>>> The only question is the user space interface. Two options spring to
>>> mind:
>>> a) Close to current code. Once kmem.limit_in_bytes is set to
>>>   non-RESOURCE_MAX, then kmem accounting is enabled and cannot be
>>>   disabled. Therefore the limit cannot be set to RESOURCE_MAX
>>>   thereafter. The largest value would be something like
>>>   RESOURCE_MAX-PAGE_SIZE. An admin wondering if kmem is enabled only
>>>   has to cat kmem.limit_in_bytes - if it's less than RESOURCE_MAX, then
>>>   kmem is enabled.
>>>
>>
>> If we need to choose between them, I like this better than your (b).
>> At least it is all clear, and "fix" the history problem, since it is
>> possible to look up the status of the files and figure it out.
>>
>>> b) Or, if we could introduce a separate sticky kmem.enabled file. Once
>>>   set it could not be unset. Kmem accounting would only be enabled if
>>>   kmem.enabled=1.
>>>
>>> I think (b) is clearer.
>>>
>> Depends on your definition of clearer. We had a knob for
>> kmem_independent in the beginning if you remember, and it was removed.
>> The main reason being knobs complicate minds, and we happen to have a
>> very natural signal for this. I believe the same reasoning applies here.
>
> Sounds good to me, so let's go with (a).
>
Michal, what do you think ?