

---

Subject: Re: [PATCH v5 04/14] kmem accounting basic infrastructure  
Posted by [Glauber Costa](#) on Thu, 18 Oct 2012 09:04:25 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On 10/18/2012 02:12 AM, Andrew Morton wrote:  
> On Tue, 16 Oct 2012 14:16:41 +0400  
> Glauber Costa <glommer@parallels.com> wrote:  
>  
>> This patch adds the basic infrastructure for the accounting of kernel  
>> memory. To control that, the following files are created:  
>>  
>> \* memory.kmem.usage\_in\_bytes  
>> \* memory.kmem.limit\_in\_bytes  
>> \* memory.kmem.failcnt  
>  
> gargh. "failcnt" is not a word. Who was it who first thought that  
> omitting voewls from words improves anything?  
>  
> Sigh. That pooch is already screwed and there's nothing we can do  
> about it now.  
>

Dunno =(

>> \* memory.kmem.max\_usage\_in\_bytes  
>>  
>> They have the same meaning of their user memory counterparts. They  
>> reflect the state of the "kmem" res\_counter.  
>>  
>> Per cgroup kmem memory accounting is not enabled until a limit is set  
>> for the group. Once the limit is set the accounting cannot be disabled  
>> for that group. This means that after the patch is applied, no  
>> behavioral changes exists for whoever is still using memcg to control  
>> their memory usage, until memory.kmem.limit\_in\_bytes is set for the  
>> first time.  
>>  
>> We always account to both user and kernel resource\_counters. This  
>> effectively means that an independent kernel limit is in place when the  
>> limit is set to a lower value than the user memory. A equal or higher  
>> value means that the user limit will always hit first, meaning that kmem  
>> is effectively unlimited.  
>>  
>> People who want to track kernel memory but not limit it, can set this  
>> limit to a very high number (like RESOURCE\_MAX - 1page - that no one  
>> will ever hit, or equal to the user memory)  
>>  
>>

```
>> ...
>>
>> +/* internal only representation about the status of kmem accounting. */
>> +enum {
>> + KMEM_ACCOUNTED_ACTIVE = 0, /* accounted by this cgroup itself */
>> +};
>> +
>> +#define KMEM_ACCOUNTED_MASK (1 << KMEM_ACCOUNTED_ACTIVE)
>> +
>> +#ifdef CONFIG_MEMCG_KMEM
>> +static void memcg_kmem_set_active(struct mem_cgroup *memcg)
>> +{
>> + set_bit(KMEM_ACCOUNTED_ACTIVE, &memcg->kmem_accounted);
>> +}
>> +#endif
>
> I don't think memcg_kmem_set_active() really needs to exist. It has a
> single caller and is unlikely to get any additional callers, so just
> open-code it there?
>
```

Actually they exist as a way to make everything fit in closer to 80-columns without writing the function spanning 10 lines.

I can open code them if you guys prefer.

---