Subject: Re: [PATCH 01/10] memcg: Kernel memory accounting infrastructure.
Posted by Suleiman Souhlal on Wed, 29 Feb 2012 00:37:29 GMT

View Forum Message <> Reply to Message

On Tue, Feb 28, 2012 at 5:10 AM, Glauber Costa <glommer@parallels.com> wrote:

```
> On 02/27/2012 07:58 PM, Suleiman Souhlal wrote:
>>
>> Enabled with CONFIG_CGROUP_MEM_RES_CTLR_KMEM.
>>
>> Adds the following files:
>>     - memory.kmem.independent_kmem_limit
>>     - memory.kmem.usage_in_bytes
>>     - memory.kmem.limit_in_bytes
>>
>> Signed-off-by: Suleiman Souhlal<suleiman@google.com>
>> ---
>>  mm/memcontrol.c |  121
>> +++++++++++++++++++++++++++++++++++++++++++++++++++++-
>>  1 files changed, 120 insertions(+), 1 deletions(-)
>>
>> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
>> index 228d646..11e31d6 100644
>> --- a/mm/memcontrol.c
>> +++ b/mm/memcontrol.c
>> @@ -235,6 +235,10 @@ struct mem_cgroup {
>>          */
>>         struct res_counter memsw;
>>         /*
>> +        * the counter to account for kernel memory usage.
>> +        */
>> +       struct res_counter kmem_bytes;
>> +       /*
>
> Not terribly important, but I find this name inconsistent. I like
> just kmem better.
```

I will change it.

```
>>          * Per cgroup active and inactive list, similar to the
>>          * per zone LRU lists.
>>          */
>> @@ -293,6 +297,7 @@ struct mem_cgroup {
>>  #ifdef CONFIG_INET
>>         struct tcp_memcontrol tcp_mem;
>>  #endif
>> +       int independent_kmem_limit;
>>  };
>
```

> bool ?
>
> But that said, we are now approaching some 4 or 5 selectables in the memcg
> structure. How about we turn them into flags?

The only other selectable (that is a boolean) I see is use_hierarchy.
Or do you also mean oom_lock and memsw_is_minimum?

Either way, I'll try to make them into flags.

```
>> @@ -4587,6 +4647,10 @@ static int register_kmem_files(struct cgroup *cont,
>> struct cgroup_subsys *ss)
>>  static void kmem_cgroup_destroy(struct cgroup_subsys *ss,
>>                                 struct cgroup *cont)
>>  {
>> +       struct mem_cgroup *memcg;
>> +
>> +       memcg = mem_cgroup_from_cont(cont);
>> +       BUG_ON(res_counter_read_u64(&memcg->kmem_bytes, RES_USAGE) != 0);
```

>
> That does not seem to make sense, specially if you are doing lazy creation.
> What happens if you create a cgroup, don't put any tasks into it (therefore,
> usage == 0), and then destroy it right away?
>
> Or am I missing something?

The BUG_ON will only trigger if there is any remaining kernel memory,
so the situation you describe should not be a problem.

-- Suleiman