Subject: Re: [RFD][PATCH] memcg: Move Usage at Task Move
Posted by Balbir Singh on Wed, 11 Jun 2008 13:13:38 GMT
View Forum Message <> Reply to Message

kamezawa.hiroyu@jp.fujitsu.com wrote:
> ----- Original Message -----
>> On Wed, 11 Jun 2008 13:57:34 +0530
>> Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>>
>> (snip)
>>
>>>>> 2. Don't move any usage at task move. (current implementation.)
>>>>>   Pros.
>>>>>   - no complication in the code.
>>>>>   Cons.
>>>>>   - A task's usage is chareged to wrong cgroup.
>>>>>   - Not sure, but I believe the users don't want this.
>>>> I'd say stick with this unless there a strong arguments in favour of
>>>> changing, based on concrete needs.
>>>>
>>>>> One reasone is that I think a typical usage of memory controller is
>>>>> fork()->move->exec(). (by libcg ?) and exec() will flush the all usage.
>>>> Exactly - this is a good reason *not* to implement move - because then
>>>> you drag all the usage of the middleware daemon into the new cgroup.
>>>>
>>> Yes. The other thing is that charges will eventually fade away. Please see
> the
>>> cgroup implementation of page_referenced() and mark_page_accessed(). The
>>> original group on memory pressure will drop pages that were left behind by
> a
>>> task that migrates. The new group will pick it up if referenced.
>>>
>> Hum..
>> So, it seems that some kind of "Lazy Mode"(#3 of Kamezawa-san's)
>> has been implemented already.
>>
>> But, one of the reason that I think usage should be moved
>> is to make the usage as accurate as possible, that is
>> the size of memory used by processes in the group at the moment.
>>
>> I agree that statistics is not the purpose of memcg(and swap),
>> but, IMHO, it's useful feature of memcg.
>> Administrators can know how busy or idle each groups are by it.
>>
> One more point. This kinds of lazy "drop" approach canoot works well when
> there are mlocked processes. lazy "move" approrach is better if we do in lazy
> way. And how quickly they drops depends on vm.swappiness.
>

> Anyway, I don't like complicated logic in the kernel.
> So, let's see how simple "move" can be implemented. Then, it will be just a
> trade-off problem, IMHO.
> If policy is fixed, implementation itself will not be complicated, I think.
>

I agree with you that it is a trade-off problem and we should keep move as simple as possible.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL