Subject: Re: [RFD][PATCH] memcg: Move Usage at Task Move
Posted by Balbir Singh on Wed, 11 Jun 2008 08:27:34 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On Thu, Jun 5, 2008 at 6:52 PM, KAMEZAWA Hiroyuki
> <kamezawa.hiroyu@jp.fujitsu.com> wrote:
>> Move Usage at Task Move (just an experimantal for discussion)
>> I tested this but don't think bug-free.
>>
>> In current memcg, when task moves to a new cg, the usage remains in the old cg.
>> This is considered to be not good.
>
> Is it really such a big deal if we don't transfer the page ownerships
> to the new cgroup? As this thread has shown, it's a fairly painful
> operation to support. It would be good to have some concrete examples
> of cases where this is needed.
>
>

I tend to agree with Paul. One of the reasons, I did not move charges is that
makes migration an expensive operation. Since migration is well controlled with
permissions, we assume that the node owner what he/she is doing.

>> This is a trial to move "usage" from old cg to new cg at task move.
>> Finally, you'll see the problems we have to handle are failure and rollback.
>>
>> This one's Basic algorithm is
>>
>>   0. can_attach() is called.
>>   1. count movable pages by scanning page table. isolate all pages from LRU.
>>   2. try to create enough room in new memory cgroup
>>   3. start moving page accouing
>>   4. putback pages to LRU.
>>   5. can_attach() for other cgroups are called.
>>
>> A case study.
>>
>> group_A -> limit=1G, task_X's usage= 800M.
>> group_B -> limit=1G, usage=500M.
>>
>> For moving task_X from group_A to group_B.
>> - group_B should be reclaimed or have enough room.
>>
>> While moving task_X from group_A to group_B.
>> - group_B's memory usage can be changed
>> - group_A's memory usage can be changed
>>

>> We accounts the resouce based on pages. Then, we can't move all resource
>> usage at once.
>>
>> If group_B has no more room when we've moved 700M of task_X to group_B,
>> we have to move 700M of task_X back to group_A. So I implemented roll-back.
>> But other process may use up group_A's available resource at that point.
>>
>> For avoiding that, preserve 800M in group_B before moving task_X means that
>> task_X can occupy 1600M of resource at moving. (So I don't do in this patch.)
>
> I think that pre-reserving in B would be the cleanest solution, and
> would save the need to provide rollback.
>
>> 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.

[snip]

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