

---

Subject: Re: [RFD][PATCH] memcg: Move Usage at Task Move  
Posted by [Daisuke Nishimura](#) on Tue, 10 Jun 2008 07:35:50 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Hi, Kamezawa-san.

Sorry for late reply.

On Fri, 6 Jun 2008 10:52:35 +0900, KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com> wrote:

> Move Usage at Task Move (just an experimental 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.  
>  
I agree.

> 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 accounting  
> 4. putback pages to LRU.  
> 5. can\_attach() for other cgroups are called.  
>

You isolate pages and move charges of them by can\_attach(),  
but it means that pages that are allocated between page isolation  
and moving task->cgroups remains charged to old group, right?

I think it would be better if possible to move charges by attach()  
as cpuset migrates pages by cpuset\_attach().  
But one of the problem of it is that attach() does not return  
any value, so there is no way to notify failure...

> 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 resource 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.)  
 >  
 > This patch uses Best-Effort rollback. Failure in rollback is ignored and  
 > the usage is just leaked.

>  
 If implement rollback in kernel, I think it must not fail to prevent  
 leak of usage.

How about using "charge\_force" for rollbak?

Or, instead of implementing rollback in kernel,  
 how about making user(or middle ware?) re-echo pid to rollbak  
 on failure?

> Roll-back can happen when  
 > (a) in phase 3. cannot move a page to new cgroup because of limit.  
 > (b) in phase 5. other cgourp subsys returns error in can\_attach().  
 >

Isn't rollbak needed on failure between can\_attach and attach(e.g. failure  
 on find\_css\_set, ...)?

```
> +int mem_cgroup_recharge_task(struct mem_cgroup *newcg,
> +    struct task_struct *task)
> +{
> +    (snip)
> +    /* create enough room before move */
> +    necessary = info.count * PAGE_SIZE;
> +
> +    do {
> +        spin_lock(&newcg->res.lock);
> +        if (newcg->res.limit > necessary)
> +            rc = -ENOMEM;
```

I think it should be (newcg->res.limit < necessary).

Thanks,  
 Daisuke Nishimura.

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---