
Subject: Re: [RFC][PATCH] forced uncharge for successful rmdir.
Posted by [KAMEZAWA Hiroyuki](#) on Mon, 01 Oct 2007 04:30:01 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi, thank you for review.

On Mon, 01 Oct 2007 09:46:02 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

```
> > @@ -424,17 +424,80 @@ void mem_cgroup_uncharge(struct page_cgr
> > if (atomic_dec_and_test(&pc->ref_cnt)) {
> >     page = pc->page;
> >     lock_page_cgroup(page);
> > - mem = pc->mem_cgroup;
> > - css_put(&mem->css);
> > - page_assign_page_cgroup(page, NULL);
> > - unlock_page_cgroup(page);
> > - res_counter_uncharge(&mem->res, PAGE_SIZE);
> > + pc = page_get_page_cgroup(page);
> > + if (pc) {
> > +     mem = pc->mem_cgroup;
> > +     css_put(&mem->css);
> > +     page_assign_page_cgroup(page, NULL);
> > +     unlock_page_cgroup(page);
> > +     res_counter_uncharge(&mem->res, PAGE_SIZE);
> > +     spin_lock_irqsave(&mem->lru_lock, flags);
> > +     list_del_init(&pc->lru);
> > +     spin_unlock_irqrestore(&mem->lru_lock, flags);
> > +     kfree(pc);
> > + } else
> > +     unlock_page_cgroup(page);
> > + }
> > + }
```

> This looks like a bug fix in mem_cgroup_uncharge(). Did you hit a
> condition of simultaneous free? Could we split this up into a separate
> patch please.

No, but forced-uncharge and usual unchage will have race.
"page" is linked to zone's LRU while unchage is going.

```
>
> > +/*
> > + * Uncharge pages in force. If the page is accessed again, it will be recharged by
> > + * other cgroup.
> > + *
> > + * mem->lru_lock guarantees no-race with mem_cgroup_isolate_pages()
```

```

>> + * lock_page_cgroup() -> pc = page_get_page_cgroup() guarantees no-race with
>> + * mem_cgroup_uncharge().
>> + */
>>
>> - spin_lock_irqsave(&mem->lru_lock, flags);
>> - list_del_init(&pc->lru);
>> - spin_unlock_irqrestore(&mem->lru_lock, flags);
>> - kfree(pc);
>> +static void
>> +mem_cgroup_forced_uncharge_list(struct mem_cgroup *mem, struct list_head *src)
>> +{
>> + struct page_cgroup *pc;
>> + struct page *page;
>> + int count = SWAP_CLUSTER_MAX;
>> +
>> + spin_lock(&mem->lru_lock);
>
> I think this should be spin_lock_irqsave.
>
Okay.

```

```

>> + while (!list_empty(src)) {
>> + pc = list_entry(src->prev, struct page_cgroup, lru);
>> + /* When we uncharge page, pc->page is not cleared before
>> + pc is removed from LRU. But, page->pc will be cleared. */
>
> Comment style needs fixing
>
Ahh, will fix.

```

```

>> + page = pc->page;
>> + lock_page_cgroup(page);
>> + pc = page_get_page_cgroup(page);
>> + /* check race */
>> + if (pc) {
>> + css_put(&mem->css);
>> + page_assign_page_cgroup(page, NULL);
>> + unlock_page_cgroup(page);
>> + res_counter_uncharge(&mem->res, PAGE_SIZE);
>> + list_del_init(&pc->lru);
>> + kfree(pc);
>> + } else
>> + unlock_page_cgroup(page);
>> + if (--count == 0) {
>> + spin_unlock(&mem->lru_lock);
>> + cond_resched();
>> + spin_lock(&mem->lru_lock);
>> + count = SWAP_CLUSTER_MAX;

```

```
>> + }
>> + }
>> + spin_unlock(&mem->lru_lock);
>> +}
>
```

> The forced_uncharge_list is one way of doing it, the other
> way is to use a shrink_all_memory() logic. For now, I think
> this should be fine.

I have both versions. I myself think forced-uncharge is better.

```
>
>> +
>> +int mem_cgroup_forced_uncharge_all(struct mem_cgroup *mem)
>> +{
>> + int ret = -EBUSY;
>> + css_get(&mem->css);
>> + while (!list_empty(&mem->active_list) ||
>> +      !list_empty(&mem->inactive_list)) {
>> + if (atomic_read(&mem->css.cgroup->count) > 0)
>> + goto out;
>> + mem_cgroup_forced_uncharge_list(mem, &mem->active_list);
>> + mem_cgroup_forced_uncharge_list(mem, &mem->inactive_list);
>> + }
>> + ret = 0;
>> +out:
>> + css_put(&mem->css);
>> + return ret;
>> + }
>>
>> int mem_cgroup_write_strategy(char *buf, unsigned long long *tmp)
>> @@ -494,7 +557,17 @@ static ssize_t mem_control_type_write(st
>> if (*end != '\0')
>> goto out_free;
>>
>> - if (tmp <= MEM_CGROUP_TYPE_UNSPEC || tmp >= MEM_CGROUP_TYPE_MAX)
>> + if (tmp == MEM_CGROUP_TYPE_UNSPEC) {
>> + if (atomic_read(&mem->css.cgroup->count) == 0) /* uncharge all */
>> + ret = mem_cgroup_forced_uncharge_all(mem);
>> + else
>> + ret = -EBUSY;
>> + if (!ret)
>> + ret = nbytes;
>> + goto out_free;
>> + }
>> +
>
```

> Can we use a different file for this? Something like
> memory.force_reclaim or memory.force_out_memory?

Yes, okay. How about drop_charge ?
(This uncharge doesn't drop memory...)

```
>  
> > + if (tmp < MEM_CGROUP_TYPE_UNSPEC || tmp >= MEM_CGROUP_TYPE_MAX)  
> >     goto out_free;  
> >  
> > mem->control_type = tmp;  
> >  
>  
> Overall, the patch looks good. I am going to stress test this patch.  
>  
Thanks. I'll post again when the next -mm comes.
```

Regards,
-Kame

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