Subject: Re: [BUGFIX][RFC][PATCH][only -mm] FIX memory leak in memory cgroup vs. page migration [1/1] fix pag
Posted by KAMEZAWA Hiroyuki on Wed, 03 Oct 2007 00:53:09 GMT
View Forum Message <> Reply to Message

On Tue, 02 Oct 2007 21:04:40 +0530
Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> KAMEZAWA Hiroyuki wrote:
> > While using memory control cgroup, page-migration under it works as following.
> > ==
> > 1. uncharge all refs at try to unmap.
> > 2. charge regs again remove_migration_ptes()
> > ==
> > This is simple but has following problems.
> > ==
> > The page is uncharged and chaged back again if *mapped*.
> >   - This means that cgroup before migraion can be different from one after
> >   migraion
>
> >From the test case mentioned earlier, this happens because the task has
> moved from one cgroup to another, right?
Ah, yes.

> > And migration can migrate *not mapped* pages in future by migration-by-kernel
> > driven by memory-unplug and defragment-by-migration at el.
> >
> > This patch tries to keep memory cgroup at page migration by increasing
> > one refcnt during it. 3 functions are added.
> > mem_cgroup_prepare_migration() --- increase refcnt of page->page_cgroup
> > mem_cgroup_end_migration()   --- decrease refcnt of page->page_cgroup
> > mem_cgroup_page_migration() --- copy page->page_cgroup from old page to
> >   new page.
> >
> > Obviously, mem_cgroup_isolate_pages() and this page migration, which
> > copies page_cgroup from old page to new page, has race.
> >
> > There seem to be 3 ways for avoiding this race.
> > A. take mem_group->lock while mem_cgroup_page_migration().
> > B. isolate pc from mem_cgroup's LRU when we isolate page from zone's LRU.
> > C. ignore non-LRU page at mem_cgroup_isolate_pages().
> >
> > This patch uses method (C.) and modifes mem_cgroup_isolate_pages() igonres
> > !PageLRU pages.
> >
>
> The page(s) is(are) !PageLRU only during page migration right?

>
Hmm...!PageLRU() means that page is not on LRU.
Then, kswapd can remove a page from LRU.

> > - if (page_zone(page) != z)
> > + if (page_zone(page) != z || !PageLRU(page)) {
>
> I would prefer to do unlikely(!PageLRU(page)), since most of the
> times the page is not under migration
>
I see.

> > +   /* Skip this */
> > +   /* Don't decrease scan here for avoiding dead lock */
>
> Could we merge the two comments to one block comment?
>
will do

> >   continue;
> > + }
> >
> >   /*
> >   * Check if the meta page went away from under us
> > @@ -417,8 +424,14 @@ void mem_cgroup_uncharge(struct page_cgr
> >   return;
> >
> >   if (atomic_dec_and_test(&pc->ref_cnt)) {
> > +retry:
> >   page = pc->page;
> >   lock_page_cgroup(page);
> > + /* migration occur ? */
> > + if (page_get_page_cgroup(page) != pc) {
> > +   unlock_page_cgroup(page);
> > +   goto retry;
>
> Shouldn't we check if page_get_page_cgroup(page) returns
> NULL, if so, unlock and return?
Hmm, I think page_get_page_cgroup(page) != pc covers it. pc is not NULL.

Thanks,
-Kame