Subject: Re: [PATCH v4 02/11] memcg: document cgroup dirty memory interfaces
Posted by Wu Fengguang on Sat, 30 Oct 2010 03:02:49 GMT
View Forum Message <> Reply to Message

On Sat, Oct 30, 2010 at 05:35:50AM +0800, Greg Thelen wrote:
> >> +A cgroup may contain more dirty memory than its dirty limit. This is possible
> >> +because of the principle that the first cgroup to touch a page is charged for
> >> +it. Subsequent page counting events (dirty, writeback, nfs_unstable) are also
> >> +counted to the originally charged cgroup.
> >> +
> >> +Example: If page is allocated by a cgroup A task, then the page is charged to
> >> +cgroup A. If the page is later dirtied by a task in cgroup B, then the cgroup A
> >> +dirty count will be incremented. If cgroup A is over its dirty limit but cgroup
> >> +B is not, then dirtying a cgroup A page from a cgroup B task may push cgroup A
> >> +over its dirty limit without throttling the dirtying cgroup B task.
> >
> > It's good to document the above "misbehavior". But why not throttling
> > the dirtying cgroup B task? Is it simply not implemented or makes no
> > sense to do so at all?
>
> Ideally cgroup B would be throttled. Note, even with this misbehavior,
> the system dirty limit will keep cgroup B from exceeding system-wide
> limits.

Yeah. And I'm OK with the current behavior, since
1) it does not impact the global limits
2) the common memcg usage (the workload you cared) seems don't share
   pages between memcg's a lot

So I'm OK to improve it in future when there comes a need.

> The challenge here is that when the current system increments dirty
> counters using account_page_dirtied() which does not immediately check
> against dirty limits. Later balance_dirty_pages() checks to see if any
> limits were exceeded, but only after a batch of pages may have been
> dirtied. The task may have written many pages in many different memcg.
> So checking all possible memcg that may have been written in the mapping
> may be a large set. I do not like this approach.

Me too.

> memcontrol.c can easily detect when memcg other than the current task's
> memcg is charged for a dirty page. It does not record this today, but
> it could. When such a foreign page dirty event occurs the associated
> memcg could be linked into the dirtying address_space so that
> balance_dirty_pages() could check the limits of all foreign memcg. In
> the common case I think the task is dirtying pages that have been
> charged to the task's cgroup, so the address_space's foreign_memcg list

> would be empty. But when such foreign memcg are dirtied
> balance_dirty_pages() would have access to references to all memcg that
> need dirty limits checking. This approach might work. Comments?

It still introduce complexities of maintaining the foreign memcg <=>
task mutual links.

Another approach may to add a parameter "struct page *page" to
balance_dirty_pages(). Then balance_dirty_pages() can check the memcg
that is associated with the _current_ dirtied page. It may not catch
all foreign memcg's, but should work fine with good probability
without introducing new data structure.

Thanks,
Fengguang

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs