Subject: Re: [RFC][PATCH] another swap controller for cgroup
Posted by yamamoto on Wed, 14 May 2008 03:21:25 GMT
View Forum Message <> Reply to Message

> >>> + BUG_ON(mm == NULL);
> >>> + BUG_ON(mm->swap_cgroup == NULL);
> >>> + if (scg == NULL) {
> >>> + /*
> >>> +   * see swap_cgroup_attach.
> >>> +   */
> >>> + smp_rmb();
> >>> + scg = mm->swap_cgroup;
> >> With the mm->owner patches, we need not maintain a separate
> >> mm->swap_cgroup.
> >
> > i don't think the mm->owner patch, at least with the current form,
> > can replace it.
> >
>
> Could you please mention what the limitations are? We could get those fixed or
> take another serious look at the mm->owner patches.

for example, its callback can't sleep.

> >>> + /*
> >>> + * swap_cgroup_attach is in progress.
> >>> + */
> >>> +
> >>> + res_counter_charge_force(&newscg->scg_counter, PAGE_CACHE_SIZE);
> >> So, we force the counter to go over limit?
> >
> > yes.
> >
> > as newscg != oldscg here means the task is being moved between cgroups,
> > this instance of res_counter_charge_force should not matter much.
> >
>
> Isn't it bad to force a group to go over it's limit due to migration?

we don't have many choices as far as ->attach can't fail.
although we can have racy checks in ->can_attach, i'm not happy with it.

> >> We need to write actual numbers here? Can't we keep the write
> >> interface consistent with the memory controller?
> >
> > i'm not sure what you mean here. can you explain a bit more?
> > do you mean K, M, etc?
> >

>
> Yes, I mean the same format that memparse() uses.

i'll take a look.

> >> Is moving to init_mm (root
> >> cgroup) a good idea? Ideally with support for hierarchies, if we ever
> >> do move things, it should be to the parent cgroup.
> >
> > i chose init_mm because there seemed to be no consensus about
> > cgroup hierarchy semantics.
> >
>
> I would suggest that we fail deletion of a group for now. I have a set of
> patches for the cgroup hierarchy semantics. I think the parent is the best place
> to move it.

ok.

> >>> + info->swap_cgroup = newscg;
> >>> + res_counter_uncharge(&oldscg->scg_counter, bytes);
> >>> + res_counter_charge_force(&newscg->scg_counter, bytes);
> >> I don't like the excessive use of res_counter_charge_force(), it seems
> >> like we might end up bypassing the controller all together. I would
> >> rather fail the destroy operation if the charge fails.
> >
> >>> + bytes = swslots * PAGE_CACHE_SIZE;
> >>> + res_counter_uncharge(&oldscg->scg_counter, bytes);
> >>> + /*
> >>> + * XXX ignore newscg's limit because cgroup ->attach method can't fail.
> >>> + */
> >>> + res_counter_charge_force(&newscg->scg_counter, bytes);
> >> But in the future, we could plan on making attach fail (I have a
> >> requirement for it). Again, I don't like the _force operation
> >
> > allowing these operations fail implies to have code to back out
> > partial operations. i'm afraid that it will be too complex.
> >
>
> OK, we need to find out a way to fix that then.

note that a failure can affect other subsystems which belong to
the same hierarchy as well, and, even worse, a back-out attempt can also fail.
i'm afraid that we need to play some kind of transaction-commit game,
which can make subsystems too complex to implement properly.

YAMAMOTO Takashi

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