Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Fri, 29 Sep 2006 00:22:59 GMT
View Forum Message <> Reply to Message

On Fri, 2006-09-29 at 03:23 +0530, Balbir Singh wrote:
> Rohit Seth wrote:
> > On Thu, 2006-09-28 at 13:31 +0530, Balbir Singh wrote:

> >
> >> (b) The other option is to do what the resource group memory controller does -
> >> build a per group LRU list of pages (active, inactive) and reclaim
> >> them using the existing code (by passing the correct container pointer,
> >> instead of the zone pointer). One disadvantage of this approach is that
> >> the global reclaim is impacted as the global LRU list is broken. At the
> >> expense of another list, we could maintain two lists, global LRU and
> >> container LRU lists. Depending on the context of the reclaim - (container
> >> over limit, memory pressure) we could update/manipulate both lists.
> >> This approach is definitely very expensive.
> >>
> >
> > Two LRUs is a nice idea. Though I don't think it will go too far. It
> > will involve adding another list pointers in the page structure. I
> > agree that the mem handler is not optimal at all but I don't want to
> > make it mimic kernel reclaimer at the same time.
>
> One possible solution is to move the container tracking out of the pages and
> into address_space and anon_vma. I guess this functionality will complicate
> task migration and accounting a bit though.
>

In the next version, I'm removing the per page pointer for container.
address_space already has a container pointer, I'm adding a pointer in
anon_vma as well. And that does seem to be complicating the accounting
just a wee bit. Though on its own, it is not helping the reclaim part.

I'll have to see how to handle kernel pages w/o a per page pointer.

> >
> >> 2. Comments on task migration support
> >>
> >> (a) One of the issues I found while using the container code is that, one could
> >> add a task to a container say "a". "a" gets charged for the tasks usage,
> >> when the same task moves to a different container say "b", when the task
> >> exits, the credit goes to "b" and "a" remains indefinitely charged.
> >>
> > hmm, when the task is removed from "a" then "a" gets the credits for the
> > amount of anon memory that is used by the task. Or do you mean
> > something different.

>
> Aah, I see. Once possible minor concern here is that a task could hope across
> several containers, it could map files in each container and allocate page
> cache pages, when it reaches the limit, it could hop to another container
> and carry on until it hits the limit there.
>
If there are multiple containers that a process can hop to then yes that
will happen.

-rohit
