Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Thu, 28 Sep 2006 18:12:19 GMT
View Forum Message <> Reply to Message

On Wed, 2006-09-27 at 15:24 -0700, Chandra Seetharaman wrote:
> On Wed, 2006-09-27 at 14:28 -0700, Rohit Seth wrote:
>
> Rohit,
>
> For 1-4, I understand the rationale. But, your implementation deviates
> from the current behavior of the VM subsystem which could affect the
> ability of these patches getting into mainline.
>

I agree that this implementation differs from existing VM subsystem.
But the key point here is, it puts the pages that should be reclaimed.
And this part needs further refining.

> IMO, the current behavior in terms of reclamation, LRU, vm_swappiness,
> and writeback logic should be maintained.
>

How? I don't want to duplicate the whole logic for containers.

> > On Wed, 2006-09-27 at 12:50 -0700, Chandra Seetharaman wrote:
> > > Rohit,
> > >
> > > I finally looked into your memory controller patches. Here are some of
> > > the issues I see:
> > > (All points below are in the context of page limit of containers being
> > > hit and the new code starts freeing up pages)
> > >
> > > 1. LRU is ignored totally thereby thrashing the working set (as pointed
> > > by Peter Zijlstra).
> >
> > As the container goes over the limit, this algorithm deactivates some of
> > the pages. I agree that the logic to find out the correct pages to
> > deactivate needs to be improved. But the idea is that these pages go in
> > front of inactive list so that if there is any memory pressure system
> > wide then these pages can easily be reclaimed.
> >
> > > 2. Frees up file pages first when hitting the page limit thereby making
> > > vm_swappiness ineffective.
> >
> > Not sure if I understood this part correctly. But the choice when the
> > container goes over its limit is between swap out some of the anonymous
> > memory first or writeback some of the dirty file pages belonging to this
> > container.

> >
> > > 3. Starts writing back pages when the # of file pages is close to the
> > >   limit, thereby breaking the current writeback algorithm/logic.
> >
> > That is done so as to ensure processes belonging to container (Whose
> > limit is hit) are the first ones getting penalized. For example, if you
> > run a tar in a container with 100MB limit then the dirty file pages will
> > be written back to disk when 100MB limit is hit). Though I will be
> > adding a HARD_LIMIT on page cache flag and the strict limit will be only
> > maintained if this container flag is set.
> >
> > > 4. MAPPED files are not counted against the page limit. why ?. This
> > >   affects reclamation behavior and makes vm_swappiness ineffective.
> >
> > num_mapped_pages only indicates how many page cache pages are mapped in
> > user page tables. More of an accounting variable.
>
> But, # of mapped pages is used in the reclamation path logic. These set
> of patches doesn't take them into account.
>
> >
> > > 5. Starts freeing up pages from the first task or the first file in the
> > >   linked list. This logic unfairly penalizes the early members of the
> > >   list.
> >
> > This is the part that I've to fix. Some per container variables that
> > remembers the last values will help here.
>
> Yes, that will help in fairness between the items in the list.
>
> But, it will still suffer from (1) above, as we would have no idea of
> the current working set (LRU) (within an item or among the items).
>

Please let me know how do you propose to have another LRU for pages in
containers. Though I can add some heuristics.

> >
> > > 6. Both active and inactive pages use physical pages. But, the
> > >   controller only counts active pages and not inactive pages. why ?
> >
> > The thought is, it is okay for containers to go over its limit as long
>
> Real number of "physical pages" used by the container is the sum of
> active and inactive pages.
>

>From the user pov, the real sum of pages that are used by container for

user land is anon + file. Now some times it is possible that there are
active pages that are neither in page cache nor in use as anon.

> My question is, shouldn't that be used to check against page limit
> instead of active pages alone ?
I can use active+inactive as the test. Sure. But I will have to also
still have a check to make sure that number of active pages themselves
is not bigger than page_limit.

> How do we describe "page limit" as (to the user) ?
>

Amount of memory below which no container throttling will happen. And
if the system is properly configured that it also ensures that this much
memory will always be there to user. If a container goes over this
limit then it will be throttled and it will suffer performance.

> > as there is enough memory in the system. When there is any memory
> > pressure then the inactive (+ dereferenced) pages get swapped out thus
> > penalizing the container. I'm also thinking of having hard limit for
>
> Reclamation goes through active pages and page cache pages before it
> gets into inactive pages. So, this may not work as you are explaining.

That is a good point. I'll have to make a check in reclaim so that when
the system is ready for swap or write back then containers are looked
first.

>
> > anonymous pages beyond which the container will not be able to grow its
> > anonymous pages.
>
> You might break the current behavior (memory pressure must be very high
> before these starts failing) if you are going to be strict about it.
>

That feature when implemented will be a container specific.

> >
> > > 7. Page limit is checked against the sum of (anon and file pages) in
> > >   some places and against active pages at some other places. IMO, it
> > >   should be always compared to the same value.
> > >
> > It is checked against sum of anon+file pages at the time when new pages
>
> why can't we check against active pages here ?
>

> > is getting allocated. But as the reclaimer activate the pages, so it is
> > also important to make sure the number of active pages is not going
> > above its limit.
>
> My point is that they won't be same (ever) and hence the check is
> inconsistent.
>

The check ensures
1- when a new page is getting added then the total sum of pages is checked against the limit.
2- Number of active pages don't exceed the limit.

These two points combined together enforce the decision that once the container goes over the limit, we scan the pages again to deactivate the excess.

Thanks,
-rohit
