Subject: Re: [ckrm-tech] [patch00/05]: Containers(V2)- Introduction
Posted by Rohit Seth on Thu, 28 Sep 2006 21:38:36 GMT
View Forum Message <> Reply to Message

On Thu, 2006-09-28 at 13:23 -0700, Chandra Seetharaman wrote:
> On Thu, 2006-09-28 at 11:12 -0700, Rohit Seth wrote:
> > 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.
>
> But, you are putting the pages up for reclamation without any
> consideration to the working set and system memory pressure.
>

And I agree with you that some heuristics need to be put there to make
that algorithm go better.

> > 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.
>
> We don't have to be duplicating the whole logic. Just make sure that the
> existing mechanisms are aware of containers, if they exist.
>

The next version is going to have hooks in kernel reclaim path for
containers. But that will still not make it close to what normal
reclaim path does for pages outside containers.

> <snip>
>
> > >
> > > 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.
>
> There are multiple ways as Balbir pointed in his email:
> - reclamation per container (as in current RG implementation)
>   (+ do a system wide reclaim when the system pressure is high)
> - reclaim with the knowledge of containers that are over limit
>   (Dave Hansen's patches + avoid overhead of combing the list)
> - have two lists one for the system and one per container
>

I will look at Dave's patch. Having two different list is not the right
approach. I will add some reclaim logic in kernel reclaim path.

Any idea why current RG implementation is not in mainline? Any effort
in reviving that and getting it in Andrew's tree.

> >
> > > >
> > > > > 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
>
> But, from the implementation one cannot clearly derive what we mean by
> "memory" here (physical ?, file + anon ?; if we say physical, it is not
> correct).

>

It is mostly correct. i.e. anon+file == total user physical memory for
container. Except for the corner cases when there could be stale
pagecache pages that are no longer on page cache but still on LRU (not
yet recalimed).

> > 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.
>
> But, the user's expectation would be that we would be throwing out pages
> based on LRU (within that container). But this implementation doesn't
> provide that behavior. It doesn't care about the working set.
>

IMO, user is expected to live inside the limits when containers are
defined. If the limits are exceeded then some performance impact will
happen. Having said that though I would still like to get some
optimizations in memory handler so that more appropriate pages are
deactivated.

> Performance impact will be lesser if we consider the working set and
> throw out pages based on LRU (within a container).
>
I don't deny it. But two separate LRUs is not an option.

> >
> > > > 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.

>
> My point is, even though it is container specific, the behavior (inside
> a container) should be same as what a user sees at the system level now.
>
> For example, consider a workload that is run on a 1G system now, and
> user sees only occasional memory allocation failures and just a handful
> of oom kills. When the workload is moved to a container with 1G,
> failures the user see should be in the same order (and similar with
> performance characteristics).
>
> Do you agree that it will be the user's expectation ?
>

That will be nice to have feature. And for that any container
implementation will have to be as tightly intertwined with rest of vm as
cpuset is.

thanks,
-rohit
>