Subject: Re: BC: resource beancounters (v2)
Posted by Rohit Seth on Sat, 26 Aug 2006 02:15:52 GMT
View Forum Message <> Reply to Message

On Fri, 2006-08-25 at 20:30 +0400, Andrey Savochkin wrote:
> On Fri, Aug 25, 2006 at 07:30:03AM -0700, Andrew Morton wrote:
> > On Fri, 25 Aug 2006 15:49:15 +0400
> > Kirill Korotaev <dev@sw.ru> wrote:
> >
> > > Andrey Savochkin wrote already a brief summary on vm resource management:
> > >
> > > ---------- cut ----------
> > > The task of limiting a container to 4.5GB of memory bottles down to the
> > > question: what to do when the container starts to use more than assigned
> > > 4.5GB of memory?
> > >
> > > At this moment there are only 3 viable alternatives.
> > >
> > > A) Have separate memory management for each container,
> > >   with separate buddy allocator, lru lists, page replacement mechanism.
> > >   That implies a considerable overhead, and the main challenge there
> > >   is sharing of pages between these separate memory managers.
> > >

Yes, sharing of pages across different containers/managers will be a
problem. Why not just disallow that scenario (that is what fake nodes
proposal would also end up doing).

> > > B) Return errors on extension of mappings, but not on page faults, where
> > >   memory is actually consumed.
> > >   In this case it makes sense to take into account not only the size of used
> > >   memory, but the size of created mappings as well.
> > >   This is approximately what "privvmpages" accounting/limiting provides in
> > >   UBC.
> > >

Keeping a tab on all the virtual mappings in a container must also be
troublesome. And IMO is not the right way to go...this is even a
stricter version of overcommit_memory...right?

> > > C) Rely on OOM killer.
> > >   This is a fall-back method in UBC, for the case "privvmpages" limits
> > >   still leave the possibility to overload the system.
> > >
> >
> > D) Virtual scan of mm's in the over-limit container
> >

This seems like an interesting choice. If we can quickly inactivate
some pages belonging to tasks in over_the_limit container.

> > E) Modify existing physical scanner to be able to skip pages which
> >   belong to not-over-limit containers.
>
> I've actually tried (E), but it didn't work as I wished.
>
> It didn't handle well shared pages.
> Then, in my experiments such modified scanner was unable to regulate
> quality-of-service. When I ran 2 over-the-limit containers, they worked
> equally slow regardless of their limits and work set size.
> That is, I didn't observe a smooth transition "under limit, maximum
> performance" to "slightly over limit, a bit reduced performance" to
> "significantly over limit, poor performance". Neither did I see any fairness
> in how containers got penalized for exceeding their limits.
>

That sure is an interesting observation though I think it really depends
on if you are doing the same amount of work when counts have just gone
above the limits to the point where they are way over the limit.

> My explanation of what I observed is that
> - since filesystem caches play a huge role in performance, page scanner will
>   be very limited in controlling container's performance if caches
>   stay shared between containers,

Yeah, if a page is shared between containers then you can end up doing
nothing useful. And that is where containers dedicated to individual
filesystem could be useful.

> - in the absence of decent disk I/O manager, stalls due to swapin/swapout
>   are more influenced by disk subsystem than by page scanner policy.
> So in fact modified page scanner provides control over memory usage only as
> "stay under limits or die", and doesn't show many advantages over (B) or (C).
> At the same time, skipping pages visibly penalizes "good citizens", not only
> in disk bandwidth but in CPU overhead as well.
>

Sure that CPU, disk and other variables will kick in when you start
swapping. But then apps are expected to suffer when gone over limit.
The drawback is the apps that are not hit the limit will also suffer,
but then that is where extra controllers like CPU will kick in.

Maybe, we have a flag for each container indicating whether the tasks
belonging to that container should be killed immediately or they are
okay to run with lower performance as far as they can.

-rohit