Subject: Re: BC: resource beancounters (v2)
Posted by Chandra Seetharaman on Fri, 25 Aug 2006 19:00:05 GMT
View Forum Message <> Reply to Message

Have you seen/tried the memory controller in CKRM/Resource Groups ?
http://sourceforge.net/projects/ckrm

It maintains a per resource group LRU lists and also maintains a list of
over-guarantee groups (with ordering based on where they are in their
guarantee-limit scale). So, when a reclaim needs to happen, pages are
first freed from a group that is way over its limit, and then the next
one and so on.

Few things that it does that are not good:
- doesn't account shared pages accurately
- moves all pages from a task when the task moves to a different group
- totally new reclamation path

regards,

chandra
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.
> > >
> > > 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.
> > >
> > > 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
> >
> > 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.
>
> 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,
> - 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.
>
> So I settled for (A)-(C) for now.
> But it certainly would be interesting to hear if someone else makes such
> experiments.
>
> Best regards
>
> Andrey
--

---------- ----------
Chandra Seetharaman | Be careful what you choose....
- sekharan@us.ibm.com | .....you may get it.
---------- ----------