Subject: Re: [PATCH] BC: resource beancounters (v2)
Posted by dev on Fri, 25 Aug 2006 11:47:28 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
>>As the first step we want to propose for discussion
>>the most complicated parts of resource management:
>>kernel memory and virtual memory.
>
>
> The patches look reasonable to me - mergeable after updating them for
> today's batch of review commentlets.
sure. will do updates as long as there are reasonable comments.

> I have two high-level problems though.
>
> a) I don't yet have a sense of whether this implementation
>   is appropriate/sufficient for the various other
>   applications which people are working on.
>
>   If the general shape is OK and we think this
>   implementation can be grown into one which everyone can
>   use then fine.
>
> And...
>
>
>>The patch set to be sent provides core for BC and
>>management of kernel memory only. Virtual memory
>>management will be sent in a couple of days.
>
>
> We need to go over this work before we can commit to the BC
> core. Last time I looked at the VM accounting patch it
> seemed rather unpleasing from a maintainability POV.
hmmm... in which regard?

> And, if I understand it correctly, the only response to a job
> going over its VM limits is to kill it, rather than trimming
> it. Which sounds like a big problem?
No, UBC virtual memory management refuses occur on mmap()'s.
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.

It would be nice, indeed, to invent something new.
The ideal mechanism would
- slow down the container over-using memory, to signal the user that
he is over his limits,
- at the same time this slowdown shouldn't lead to the increase of memory
usage: for example, a simple slowdown of apache web server would lead
to the growth of the number of serving children and consumption of more
memory while showing worse performance,
- and, at the same time, it shouldn't penalize the rest of the system from
the performance point of view...

May be this can be achieved via carefully tuned swapout mechanism together
with disk bandwidth management capable of tracking asynchronous write
requests, may be something else is required.
It's really a big challenge.

Meanwhile, I guess we can only make small steps in improving Linux resource
management features for this moment.
---------- cut ----------

Thanks,
Kirill