Subject: Re: [PATCH] BC: resource beancounters (v2)
Posted by dev on Tue, 29 Aug 2006 15:33:04 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Fri, 25 Aug 2006 15:49:15 +0400
> Kirill Korotaev <dev@sw.ru> wrote:
>
>
>>>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?
>
>
> Little changes all over the MM code which might get accidentally broken.
>
>
>>>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.
>
>
> That's worse, isn't it? Firstly it rules out big sparse mappings and secondly
1) if mappings are private then yes, you can not mmap too much. This is logical,
since this whole mappings are potentially allocatable and there is no way to control
it later except for SIGKILL.
2) if mappings are shared file mappings (shmem is handled in similar way) then
  you can mmap as much as you want, since these pages can be reclaimed.

> mmap_and_use(80% of container size)
> fork_and_immediately_exec(/bin/true)
>
> will fail at the fork?
yes, it will fail on fork() or exec() in case of much private (1) mappings.
fail on fork() and exec() is much friendlier then SIGKILL, don't you think so?

private mappings parameter which is limited by UBC is a kind of upper estimation
of container RSS. From our experience such an estimation is ~ 5-20% higher then
a real physical memory used (with real-life applications).

>>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.
>
> F) Something else ;)
We fully agree that other possible algorithms can and should exist.
My idea only is that any of them would need accounting anyway
(which is the most part of beancounters).
Throtling, modified scanners etc. can be implemented as a separate
BC parameters. Thus, an administrator will be able to select
which policy should be applied to the container which is near its limit.

So the patches I'm trying to send are a step-by-step accounting of all
the resources and their simple limitations. More comprehensive limitation
policy will be built on top of it later.

BTW, UBC page beancounters allow to distinguish pages used by only one
container and pages which are shared. So scanner can try to reclaim
container private pages first, thus not influencing other containers.

Thanks,
Kirill