Subject: Re: Re: [ckrm-tech] containers development plans (July 10 version)
Posted by Takenori Nagano on Wed, 11 Jul 2007 12:18:25 GMT

Hi,

I think Balbir's idea is very simple and reasonable way to develop per container
swapping. Because kernel needs the information that a target page belongs to
which container. Fortunately, we already had page based memory management system
which included in RSS controller. I think it is appropriate that we develop per
container swapping on page based memory management system.

I feel better Balbir's approach.

Balbir Singh wrote:
> Paul Menage wrote:
>> On 7/11/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>>> swap_list is a list of swap_devices associated with the container.
>> That doesn't sound so great, since you'd need to update all the
>> mem_container_ptr objects that point to that swap controller subsys
>> state when you change the swap devices for the container.
>>
>
> Not all of them, only for that container. This list is per container.
> I don't see why need to update all the mem_container_ptr objects?
>
>>>> - when an mm is created, store a pointer to the task_struct that it
>>>> belongs to
>>>> - when a process exits and its mm_struct points to it, and there are
>>>> other mm users (i.e. a thread group leader exits before some of its
>>>> children), then find a different process that's using the same mm
>>>> (which will almost always be the next process in the list running
>>>> through current->tasks, but in strange situations we might need to
>>>> scan the global tasklist)
>>>>
>>> We'll that sounds like a complicated scheme.
>> I don't think it's that complicated. There would be some slightly
>> interesting synchronization, probably involving RCU, to make sure you
>> didn't derefence mm->owner when mm->owner had been freed but apart
>> from that it's straightforward.
>>
>
> Walking the global tasklist to find the tasks that share the same mm
> to me seems like an overhead.
>
>>> We do that currently, our mm->owner is called mm->mem_container.
>> No.
>>

>> mm->mem_container is a pointer to a container (well, actually a
>> container_subsys_state). As Pavel mentioned in my containers talk,
>> giving non-task objects pointers to container_subsys_state objects is
>> possible but causes problems when the actual tasks move around, and if
>> we could avoid it that would be great.
>>
>
> Hmmm.. interesting.. I was there, but I guess I missed the discussion
> (did u have it after the talk?)
>
>>> It points
>>> to a data structure that contains information about the container to which
>>> the mm belongs. The problem I see with mm->owner is that several threads
>>> can belong to different containers.
>> Yes, different threads could be in different containers, but the mm
>> can only belong to one container. Having it be the container of the
>> thread group leader seems quite reasonable to me.
>>
>>> I see that we probably mean the same
>>> thing, except that you suggest using a pointer to the task_struct from
>>> mm_struct, which I am against in principle, due to the complexity of
>>> changing owners frequently if the number of threads keep exiting at
>>> a rapid rate.
>> In the general case the thread group leader won't be exiting, so there
>> shouldn't be much need to update it.
>>
>
>> Paul
>>
>
>

--
Takenori Nagano <t-nagano@ah.jp.nec.com>