Subject: Re: [ckrm-tech] containers development plans (July 10 version)
Posted by Paul Menage on Wed, 11 Jul 2007 07:31:22 GMT
View Forum Message <> Reply to Message

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.

> >
> > - 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.

>
> 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.

> 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

Subject: Re: [ckrm-tech] containers development plans (July 10 version)
Posted by Balbir Singh on Wed, 11 Jul 2007 08:32:05 GMT
View Forum Message <> Reply to Message

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
>

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

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>

Subject: Re: [ckrm-tech] containers development plans (July 10 version)
Posted by Paul Menage on Wed, 11 Jul 2007 12:18:52 GMT
View Forum Message <> Reply to Message

On 7/11/07, Balbir Singh <balbir@linux.vnet.ibm.com> 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?

What if the mm is in different containers for the swap controller and
the page controller? (i.e. the two controllers were mounted on
different hierarchies, which can easily be the case if one of them is
in use, and the other isn't).

In that case you'd end up with one mem_container_ptr object for each
combination of (swap container, page container) and would basically be
reimplementing the css_group support, but for mm_struct rather than
task_struct.

And since there could be multiple mem_container_ptr objects for the
same swap controller container state, you'd need to update multiple
lists.

> >
> > 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.

As I mentioned above, I think that would be very rare, because:

1) Most tasks when they exit would either not be the mm owner (child
threads) or else there would be no other mm users (non-threaded apps)

2) If you're the mm owner and there are other users, the most likely
place to find another user of that mm would be to find the next
task_struct in your task group.

3) If there are no other tasks in your task group then one of your
siblings or children will probably be one of the other users

4) In very rare cases (not sure any come to mind right now, but maybe
if you were doing funky things with clone) you might need to walk the
global tasklist.

>
> Hmmm.. interesting.. I was there, but I guess I missed the discussion
> (did u have it after the talk?)

It was one of the questions that Pavel asked. He was asking in the context of processes changing container, and having resources left behind in the old container. It's basically the problem of when you consume non-renewable resources that aren't uniquely associated with a single task struct, what happens if some of the tasks that are responsible get moved to a different container. With a unique owner task for each mm, at least we wouldn't face this problem with mm_struct any longer, although the individual pages still have this problem.

Paul

Subject: Re: Re: [ckrm-tech] containers development plans (July 10 version)
Posted by Paul Menage on Wed, 11 Jul 2007 12:22:20 GMT
View Forum Message <> Reply to Message

On 7/11/07, Takenori Nagano <t-nagano@ah.jp.nec.com> wrote:
> 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.
>

So what about people who want to do per-container swapping without using a page-based memory controller?

Paul

Subject: Re: Re: [ckrm-tech] containers development plans (July 10 version)
Posted by serge on Wed, 11 Jul 2007 15:00:33 GMT
View Forum Message <> Reply to Message

Quoting Paul Menage (menage@google.com):
> On 7/11/07, Takenori Nagano <t-nagano@ah.jp.nec.com> wrote:
> >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.
> >
>
> So what about people who want to do per-container swapping without
> using a page-based memory controller?

Hmm, yes, I'm a few steps behind, but will what Balbir is suggesting
prevent me from just using a per-container swapfile for checkpointing
an application? (where the checkpoint consists of kicking out the
dirty pages to swap and making a new copy-on-write version of the
swapfile)

thanks,
-serge

Subject: Re: Re: [ckrm-tech] containers development plans (July 10 version)
Posted by Balbir Singh on Wed, 11 Jul 2007 17:10:09 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> Quoting Paul Menage (menage@google.com):
>> On 7/11/07, Takenori Nagano <t-nagano@ah.jp.nec.com> wrote:
>>> 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.
>>>
>> So what about people who want to do per-container swapping without
>> using a page-based memory controller?
>
> Hmm, yes, I'm a few steps behind, but will what Balbir is suggesting
> prevent me from just using a per-container swapfile for checkpointing
> an application? (where the checkpoint consists of kicking out the
> dirty pages to swap and making a new copy-on-write version of the
> swapfile)
>

I don't think it should. However, it would be nice to start this effort

right away, so that we can ensure it works both for memory control
and containers.

> thanks,
> -serge

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

Subject: Re: Re: [ckrm-tech] containers development plans (July 10 version)
Posted by Dave Hansen on Wed, 11 Jul 2007 18:04:06 GMT
View Forum Message <> Reply to Message

On Wed, 2007-07-11 at 21:18 +0900, Takenori Nagano wrote:
> 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.

There are a couple of concepts being thrown about here, so let's
separate them out a bit.

1. Limit a container's usage of swap.
   - Keep track of how many swap pages a container uses
   - go OOM on the container when it exceeds its allowed usage
   - tracking will be on a container's use of swap globally, no matter
     what swap device or file it is actually allocated in
   - all containers share all swapfiles
2. Keep separate lists of swap devices for each container
   - each container is allowed to use a subset of the system's
     swap files
   eventually:
   - keep a per-container list of which pte values correspond
     to which swapfiles
   - pte swap values are only valid inside of one container
3. Use a completely isolated set of swapfiles from (2) for
   checkpoint/restart
   - ensures that any swapfile will only contain data from one container

The idea in (1) is not very useful for checkpoint/restart, but it would
be useful to solve the cpuset OOM problem described in the VM BOF.
(That problem is basically that a cpuset with available memory but a

large amount in swap can cause another cpuset to go OOM. The memory
footprint in the system is under RAM+swap, but the OOM still happens.)

-- Dave

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: [ckrm-tech] containers development plans (July 10 version)
Posted by Herbert Poetzl on Wed, 11 Jul 2007 18:59:07 GMT
View Forum Message <> Reply to Message

On Wed, Jul 11, 2007 at 11:04:06AM -0700, Dave Hansen wrote:
> On Wed, 2007-07-11 at 21:18 +0900, Takenori Nagano wrote:
> > 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.
>
> There are a couple of concepts being thrown about here, so let's
> separate them out a bit.
>
> 1. Limit a container's usage of swap.
> - Keep track of how many swap pages a container uses
> - go OOM on the container when it exceeds its allowed usage
> - tracking will be on a container's use of swap globally, no matter
> what swap device or file it is actually allocated in
> - all containers share all swapfiles

this is probably what Linux-VServer would prefer,
but the aim would be to allow certain contexts to
keep contexts from swapping out even when over
their memory limits as long as there is enough
memory available (could be reservation or best
efford based)

> 2. Keep separate lists of swap devices for each container
> - each container is allowed to use a subset of the system's
> swap files

sounds okay too ...

> eventually:

> - keep a per-container list of which pte values correspond
> to which swapfiles
> - pte swap values are only valid inside of one container

smells like additional memory and cpu overhead

> 3. Use a completely isolated set of swapfiles from (2) for
> checkpoint/restart
> - ensures that any swapfile will only contain data from one container
>
> The idea in (1) is not very useful for checkpoint/restart, but it would
> be useful to solve the cpuset OOM problem described in the VM BOF.
> (That problem is basically that a cpuset with available memory but a
> large amount in swap can cause another cpuset to go OOM. The memory
> footprint in the system is under RAM+swap, but the OOM still happens.)

best,
Herbert

> -- Dave
>
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: Re: [ckrm-tech] containers development plans (July 10 version)
Posted by Dave Hansen on Wed, 11 Jul 2007 19:46:01 GMT
View Forum Message <> Reply to Message

On Wed, 2007-07-11 at 20:59 +0200, Herbert Poetzl wrote:
> > 2. Keep separate lists of swap devices for each container
> > - each container is allowed to use a subset of the system's
> > swap files
>
> sounds okay too ...
>
> > eventually:
> > - keep a per-container list of which pte values correspond
> > to which swapfiles
> > - pte swap values are only valid inside of one container
>
> smells like additional memory and cpu overhead

At the point when you care what these are, you're either writing to swap
or reading from it. I don't think it's a very cpu-sensitive path.

In any case, we don't have to do this now. We'll just be limited to the
existing number of global swapfiles.

-- Dave