Subject: Re: [PATCH 0/16] Pid namespaces
Posted by Dave Hansen on Fri, 06 Jul 2007 16:26:52 GMT
View Forum Message <> Reply to Message

On Fri, 2007-07-06 at 12:01 +0400, Pavel Emelianov wrote:
> This is "submition for inclusion" of hierarchical, not kconfig
> configurable, zero overheaded ;) pid namespaces.

Pavel, I'm a bit disappointed that you went ahead and sent this. I thought that, perhaps, you might have brought up how displeased you were with Suka's patches when we discussed them at OLS.

Hold your horses there a bit. This has "little" overhead for the common case, which is a single level of pid namespaces. That means that it is quick to access the "global" pid which would be the one that the "host container" sees. It also provides quick access to the pid which a containerized task gets when the task itself calls getpid(). This quick access is provided by storing the values directly in the task struct.

However, when there is more than one level in the container hierarchy, the optimization breaks down. A process which exists in a three-level hierarchy has slow access to the middle level pid. Your approach stores this information in a linked list, and surely *that* is going to have overhead in fork().

> 2. Suka's patches have the limit of pid namespace nesting.
> My patches do not.

I wouldn't say it that bluntly. Suka's patches have a configurable limit simply because it makes the implementation simpler and faster. There was also a version which dynamically allocated structures and had no inherent limits, but this was _much_ simpler. We could add dynamic allocation to this in the future and only overflow into that case if we overrun the static buffers.

That is, in effect, what your patches do. They hard-code for a two-level container, and dynamically allocate the levels after that. Suka's patches allow for arbitrary (but, config-time fixed) depth to be optimized for, and don't disallow a future dynamically-allocated completely arbitrary depth.

All of that said, I think that your approach would probably _work_ for our needs. I agree with Eric Biederman that your approach is a bit of a hack (with the hard-coded optimization for two levels), but it would certainly _work_, or we can make it work. That said, is it possible for Suka's to work for you?

> 3. Suka assumes that pid namespace can live without proc mount

> and tries to make the code work with pid_ns->proc_mnt change
> from NULL to not-NULL from times to times.
> My code calls the kern_mount() at the namespace creation and
> thus the pid_namespace always works with proc.

Have you run this by Al Viro and the other fs guys? /proc is a weird
beast :)

-- Dave

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

---

Subject: Re: [PATCH 0/16] Pid namespaces
Posted by Pavel Emelianov on Mon, 09 Jul 2007 05:58:49 GMT
View Forum Message <> Reply to Message

Dave Hansen wrote:
> On Fri, 2007-07-06 at 12:01 +0400, Pavel Emelianov wrote:
>> This is "submition for inclusion" of hierarchical, not kconfig
>> configurable, zero overheaded ;) pid namespaces.
>
> Pavel, I'm a bit disappointed that you went ahead and sent this. I
> thought that, perhaps, you might have brought up how displeased you were
> with Suka's patches when we discussed them at OLS.
>
> Hold your horses there a bit. This has "little" overhead for the common
> case, which is a single level of pid namespaces. That means that it is
> quick to access the "global" pid which would be the one that the "host
> container" sees. It also provides quick access to the pid which a
> containerized task gets when the task itself calls getpid(). This quick
> access is provided by storing the values directly in the task struct.
>
> However, when there is more than one level in the container hierarchy,
> the optimization breaks down. A process which exists in a three-level
> hierarchy has slow access to the middle level pid. Your approach stores
> this information in a linked list, and surely *that* is going to have

No. This approach stores numerical values in array. I have
removed the lists at all.

> overhead in fork().
>
>> 2. Suka's patches have the limit of pid namespace nesting.
>> My patches do not.

>
> I wouldn't say it that bluntly. Suka's patches have a configurable
> limit simply because it makes the implementation simpler and faster.

I didn't say that this difference is crucial either. I just pointed
all the major differences out. The main difference (you lost it
without any comment, but this difference is the main reason I send
my patches) is that the *approaches* differ.

> There was also a version which dynamically allocated structures and had
> no inherent limits, but this was _much_ simpler. We could add dynamic
> allocation to this in the future and only overflow into that case if we
> overrun the static buffers.
>
> That is, in effect, what your patches do. They hard-code for a
> two-level container, and dynamically allocate the levels after that.

Nope. This approach treats all the levels in a same way. My
previous version of patches had configurable flat/multilevel
models, but this patch set has no Kconfig options and makes no
difference between the 2nd and the 5th levels. However there are
some lightweight optimizations concerning the init ns. This is
done so not to affect the kernel for people who do not need the
namespaces at all.

> Suka's patches allow for arbitrary (but, config-time fixed) depth to be
> optimized for, and don't disallow a future dynamically-allocated
> completely arbitrary depth.
>
> All of that said, I think that your approach would probably _work_ for
> our needs. I agree with Eric Biederman that your approach is a bit of a
> hack (with the hard-coded optimization for two levels), but it would

Wrong again. As I have told - this set makes no difference between
levels of namespaces nesting. I have reimplemented my whole set.

> certainly _work_, or we can make it work. That said, is it possible for
> Suka's to work for you?

It may work, but as I have said there are (currently) two approaches
to make pid namespaces. This difference is described in details in
my original [PATCH 0/16] letter.

>> 3. Suka assumes that pid namespace can live without proc mount
>>   and tries to make the code work with pid_ns->proc_mnt change
>>   from NULL to not-NULL from times to times.
>>   My code calls the kern_mount() at the namespace creation and
>>   thus the pid_namespace always works with proc.

>
> Have you run this by Al Viro and the other fs guys? /proc is a weird
> beast :)

Proc changes are trivial. The main difference is that different proc
mount can have different super blocks. However this is the thing you
are right with - I had to Cc: Al Viro with the proc patch...

> -- Dave

Thanks,
Pavel

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

---

Subject: Re: [PATCH 0/16] Pid namespaces
Posted by Dave Hansen on Mon, 09 Jul 2007 19:58:37 GMT
View Forum Message <> Reply to Message

On Mon, 2007-07-09 at 09:58 +0400, Pavel Emelianov wrote:
> Dave Hansen wrote:
> > On Fri, 2007-07-06 at 12:01 +0400, Pavel Emelianov wrote:
> >> This is "submition for inclusion" of hierarchical, not kconfig
> >> configurable, zero overheaded ;) pid namespaces.
> >
> > Pavel, I'm a bit disappointed that you went ahead and sent this. I
> > thought that, perhaps, you might have brought up how displeased you were
> > with Suka's patches when we discussed them at OLS.
> >
> > Hold your horses there a bit. This has "little" overhead for the common
> > case, which is a single level of pid namespaces. That means that it is
> > quick to access the "global" pid which would be the one that the "host
> > container" sees. It also provides quick access to the pid which a
> > containerized task gets when the task itself calls getpid(). This quick
> > access is provided by storing the values directly in the task struct.
> >
> > However, when there is more than one level in the container hierarchy,
> > the optimization breaks down. A process which exists in a three-level
> > hierarchy has slow access to the middle level pid. Your approach stores
> > this information in a linked list, and surely *that* is going to have
>
> No. This approach stores numerical values in array. I have
> removed the lists at all.

Ahh. I was confused by the hlist in 'struct pid'. You are very correct.

Suka actually coded up something very, very similar to what you have. I just made him remove some of it so that the patches could be more easily reviewed. I figured we could add the fully dynamic allocation later, which you have already done. That part of your patches is remarkably similar.

-- Dave

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