Subject: Re: [PATCH 0/16] Pid namespaces
Posted by Pavel Emelianov on Mon, 09 Jul 2007 13:16:17 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Fri, Jul 06, 2007 at 12:01:59PM +0400, Pavel Emelianov wrote:
>> This is "submition for inclusion" of hierarchical, not kconfig
>> configurable, zero overheaded ;) pid namespaces.
>>
>> The overall idea is the following:
>>
>> The namespace are organized as a tree - once a task is cloned
>> with CLONE_NEWPIDS (yes, I've also switched to it :) the new
>> namespace becomes the parent's child and tasks living in the
>> parent namespace see the tasks from the new one. The numerical
>> ids are used on the kernel-user boundary, i.e. when we export
>> pid to user we show the id, that should be used to address the
>> task in question from the namespace we're exporting this id to.
>
> how does that behave when:
>
> a) the parent dies and gets reaped?

The children are re-parented to the namespace's init.
Surprised?

> b) the 'spawned' init dies, but other tasks
>   inside the pid space are still active?

The init's init becomes the namespace's init.

> c) what visibility rules do apply for the
>   various spaces (including the default host space)?

Each task sees tasks from its namespace and all its children
namespaces. Yes, each task can see itself as well ;)

>> The main difference from Suka's patches are the following:
>>
>> 0. Suka's patches change the kernel/pid.c code too heavy.
>>   This set keeps the kernel code look like it was without
>>   the patches. However, this is a minor issue. The major is:
>>
>> 1. Suka's approach is to remove the notion of the task's
>>   numerical pid from the kernel at all. The numbers are
>>   used on the kernel-user boundary or within the kernel but
>>   with the namespace this nr belongs to. This results in
>>   massive changes of struct's members fro int pid to struct

>>   pid *pid, task->pid becomes the virtual id and so on and
>>   so forth.
>>   My approach is to keep the good old logic in the kernel.
>>   The task->pid is a global and unique pid, find_pid() finds
>>   the pid by its global id and so on. The virtual ids appear
>>   on the user-kernel boundary only. Thus drivers and other
>>   kernel code may still be unaware of pids unless they do not
>>   communicate with the userspace and get/put numerical pids.
>
> interesting ... not sure that is what kernel folks
> have in mind though (IIRC, the struct pid change was
> considered a kernel side cleanup)

That's why I'm sending the patches - to make "kernel folks" make
a decision. Will we see some patches from VServer team?

>> And some more minor differences:
>>
>> 2. Suka's patches have the limit of pid namespace nesting.
>>   My patches do not.
>>
>> 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.
>
> shouldn't that be done by userspace instead?

It can be. But when the namespace is being created there's no
any userspace in it yet.

>> There are some small issues that I can describe if someone is
>> interested.
>>
>> The tests like nptl perf, unixbench spawn, getpid and others
>> didn't reveal any performance degradation in init_namespace
>> with the RHEL5 kernel .config file. I admit, that different
>> .config-s may show that patches hurt the performance, but the
>> intention was *not* to make the kernel work worse with popular
>> distributions.
>>
>> This set has some ways to move forward, but this is some kind
>> of a core, that do not change the init_pid_namespace behavior
>> (checked with LTP tests) and may require some hacking to do
>> with the namespaces only.
>
> TIA,

> Herbert
>

BTW, why did you remove Suka and Serge from Cc?

Pavel

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