Subject: Re: [PATCH 16/28] [FLAT 1/6] Changes in data structures for flat model
Posted by ebiederm on Wed, 20 Jun 2007 17:24:23 GMT
View Forum Message <> Reply to Message

Pavel Emelianov <xemul@openvz.org> writes:

> Eric W. Biederman wrote:
>> Pavel Emelianov <xemul@openvz.org> writes:
>>
>
> [snip]
>
>>>> | --- ./include/linux/sched.h.flatdatast 2007-06-15 15:14:33.000000000 +0400
>>>> | +++ ./include/linux/sched.h 2007-06-15 15:19:14.000000000 +0400
>>>> | @@ -482,7 +482,10 @@ struct signal_struct {
>>>> | pid_t session __deprecated;
>>>> | pid_t __session;
>>>> | };
>>>> | -
>>>> | +#ifdef CONFIG_PID_NS_FLAT
>>>> | + pid_t vpgrp;
>>>> | + pid_t vsession;
>>>> | +#endif
>>>> | /* boolean value for session group leader */
>>>> | int leader;
>>>> |
>>>> | @@ -944,6 +947,11 @@ struct task_struct {
>>>> | unsigned did_exec:1;
>>>> | pid_t pid;
>>>> | pid_t tgid;
>>>> | +#ifdef CONFIG_PID_NS_FLAT
>>>> | + /* hash the virtual ids as well */
>>>> | + pid_t vpid;
>>>> | + pid_t vtgid;
>>>> | +#endif
>>
>> Adding vpgrp, vsession, vpid, and vtgid is wrong.
>>
>> A case can probably be made for caching the common case (users view),
>> but we already have fields for that.
>>
>> For a global view we must use struct pid *, otherwise we are just asking
>> for trouble.
>
> Nope. If we have global unique numerical pid we're not asking for
> trouble. We're just making kernel work like it always did. Virtual
> pid makes sense *only* when interacting with user level.
>

> Making task->pid virtual is asking for trouble.

I'm not talking about making it virtual. Virtualization is the wrong concept. We aren't virtualizing something that isn't already virtualized. We are changing the implementation of an abstraction.

I'm talking about keeping task->pid as what the user space sees. Full stop.

The only point of even retaining task->pid or any of the other numeric pid identifiers on struct task_struct, struct signal_struct in data structures outside of struct pid is as an optimization. To make it clearer what we are doing we may want to rename the fields in task_struct and signal_struct but in no sense can I see use needing to cache anything except the common case which is what user space sees.

None of the implementations in the kernel needs a global numeric identifier for processes. In all cases a pointer to a struct pid is just as good from a uniqueness perspective. And all of the heavy lifting has already been done, to change the in kernel users to use struct pid pointers. There are only a few remaining cases that need to be touched now and those cases are the cases where semantically things matter.

There is very much a human interface issue with needing global numeric process ids (as single dense namespace for things is easier for people to wrap their heads around). However we have global numeric ids that live in struct pid, and show up in the init_pid_namespace. So that is not an issue.

Given this conversation I think it is time to investigate if the optimization of reading pid values from the task struct instead of struct pid is measurable.

Eric

_____

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