Subject: Pid namespaces approaches testing results
Posted by Pavel Emelianov on Tue, 29 May 2007 11:45:54 GMT
View Forum Message <> Reply to Message

Hi Eric, Suka, guys.

I have tested the following configurations:

1. 2.6.21-mm2 kernel with Suka's patches with CONFIG_PID_NS=n
2. the same with CONFIG_PID_NS=y

3. 2.6.22-rc1-mm1 kernel with my own realisation (patches will
   be sent later if interesting) with CONFIG_PID_NS=n;
4. the same with CONFIG_PID_NS=y and flat model (OpenVZ view)
   I sent earlier;
5. the same with multilevel model of my own. The difference is
   that I use hash to lookup pid_elem from struct pid/pid_t nr,
   not a plain "for" loop like in Suka's patches.

The tests run were:
1. Unixbench spawn test
2. Unixbench execl test
3. Unixbensh shell test
4. System time for ps -xaf run in a loop (1000 times)

The hardware used is 2x Intel(R) Xeon(TM) CPU 3.20GHz box with
2Gb of RAM. All the results are reproducible with 0.1% accuracy.
The slowdown is shown in comparison to the according results for
CONFIG_PID_NS=n kernel.

Summary:
Suka's model gives us about 1.5% of overhead.
My multilevel model gives us about 0.7% of overhead.
My flat model gives us an overhead comparative to
the accuracy of the measurement, i.e. zero overhead.

The detailed results are the following:

Test name:	spawn	execl	shell	ps (sys time)
1(no ns) :	579.1	618.3	1623.2	3.052s
2(suka's):	570.7	610.8	1600.2	3.107s
Slowdown :	1.5%	1.3%	1.4%	1.8%
3(no ns) :	580.6	616.0	1633.8	3.050s
4(flat) :	580.8	615.1	1632.2	3.054s
Slowdown :	0%	0.1%	<0.1%	0.1%
5(multi) :	576.9	611.0	1618.8	3.065s
Slowdown :	0.6%	0.8%	0.9%	0.5%

For the first three tests the result is better the higher the
number is. For the last test - the result is better the lower the
number is (since it is a time spent in kernel).

The results in the namespace may be worse.

If you are interested I can send my patches for pre-review and
cooperation. With the results shown I think the we do must have
the flat model as an option in the kernel for those who don't
need the infinite nesting, but cares for the kernel performance.

Thanks,
Pavel.

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

---

Subject: Re: Pid namespaces approaches testing results
Posted by serue on Wed, 30 May 2007 13:27:19 GMT
View Forum Message <> Reply to Message

Quoting Pavel Emelianov (xemul@openvz.org):
> Dave Hansen wrote:
> > On Tue, 2007-05-29 at 15:45 +0400, Pavel Emelianov wrote:
> >> The detailed results are the following:
> >> Test name: spawn execl shell ps (sys time)
> >> 1(no ns) : 579.1 618.3 1623.2 3.052s
> >> 2(suka's): 570.7 610.8 1600.2 3.107s
> >> Slowdown : 1.5% 1.3% 1.4% 1.8%
> >>
> >> 3(no ns) : 580.6 616.0 1633.8 3.050s
> >> 4(flat) : 580.8 615.1 1632.2 3.054s
> >> Slowdown : 0% 0.1% <0.1% 0.1%
> >> 5(multi) : 576.9 611.0 1618.8 3.065s
> >> Slowdown : 0.6% 0.8% 0.9% 0.5%
> >
> > Wow, thanks so much for running those. You're a step ahead of us,
> > there!
>
> Thanks :) Maybe we shall cooperate then and make three series
> of patches like
>
> 1. * The Kconfig options;
>
> * The API. I.e. calls like task_pid_nr(), task_session_nr_ns() etc;
> This part is rather important as I found that some places in kernel

> where I had to lookup the hash in multilevel model were just pid->vpid
> dereference in flat model. This is a good optimization.
>
> * The changes in the generic code that intruduce a bunch of
> #ifdef CONFIG_PID_NS
> ...
> #else
> #ifdef CONFIG_PID_NS_FLAT
> #endif
> #ifdef CONFIG_PID_NS_MULTILEVEL
> #endif
> #endif
> code in pid.c, sched.c, fork.c etc
>
> This patchset will have to make kernel prepared for namespaces injections
> and (!) not to break normal kernel operation with CONFIG_PID_NS=n.

In principle there's nothing at all wrong with that (imo). But the
thing is, given the way Suka's patchset is set up, there really isn't
any reason why it should be slower when using only one or two pid
namespaces.

Suka, right now are you allocating the struct upid separately from the
struct pid? That alone might slow things down quite a bit. By
allocating them as one large struct - saving both an alloc at clone, and
a dereference when looking at pid.upid[0] to get the pid_ns for instance
- you might get some of this perf back.

(Hmm, taking a quick look, it seems you're allocating the memory as one
chunk, but then even though the struct upid is just at the end of the
struct pid, you use a pointer to find the struct upid. That could slow
things down a bit)

Anyway, Pavel, I'd like to look at some profiling data (when Suka or I
collects some) and see whether the slowdown is fixable. If it isn't,
then we should definately look at combining the patchsets.

thanks,
-serge

> 2. The flat pid namespaces (my part)
> 3. The multilevel pid namespaces (suka's part)
>
> > Did you happen to collect any profiling information during your runs?
>
> Unfortunately no :(My intention was to prove that hierarchy has
> performance implications and should be considered carefully.
>

> > -- 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: Pid namespaces approaches testing results
Posted by Pavel Emelianov on Wed, 30 May 2007 14:03:17 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> Quoting Pavel Emelianov (xemul@openvz.org):
>> Dave Hansen wrote:
>>> On Tue, 2007-05-29 at 15:45 +0400, Pavel Emelianov wrote:
>>>> The detailed results are the following:
>>>> Test name: spawn execl shell ps (sys time)
>>>> 1(no ns) : 579.1 618.3 1623.2 3.052s
>>>> 2(suka's): 570.7 610.8 1600.2 3.107s
>>>> Slowdown : 1.5% 1.3% 1.4% 1.8%
>>>>
>>>> 3(no ns) : 580.6 616.0 1633.8 3.050s
>>>> 4(flat) : 580.8 615.1 1632.2 3.054s
>>>> Slowdown : 0% 0.1% <0.1% 0.1%
>>>> 5(multi) : 576.9 611.0 1618.8 3.065s
>>>> Slowdown : 0.6% 0.8% 0.9% 0.5%
>>> Wow, thanks so much for running those. You're a step ahead of us,
>>> there!
>> Thanks :) Maybe we shall cooperate then and make three series
>> of patches like
>>
>> 1. * The Kconfig options;
>>
>> * The API. I.e. calls like task_pid_nr(), task_session_nr_ns() etc;
>> This part is rather important as I found that some places in kernel
>> where I had to lookup the hash in multilevel model were just pid->vpid
>> dereference in flat model. This is a good optimization.
>>
>> * The changes in the generic code that intruduce a bunch of
>> #ifdef CONFIG_PID_NS
>> ...

>> #else
>> #ifdef CONFIG_PID_NS_FLAT
>> #endif
>> #ifdef CONFIG_PID_NS_MULTILEVEL
>> #endif
>> #endif
>> code in pid.c, sched.c, fork.c etc
>>
>> This patchset will have to make kernel prepared for namespaces injections
>> and (!) not to break normal kernel operation with CONFIG_PID_NS=n.
>
> In principle there's nothing at all wrong with that (imo). But the
> thing is, given the way Suka's patchset is set up, there really isn't
> any reason why it should be slower when using only one or two pid
> namespaces.

One of the main bottlenecks I see is that the routine struct_pid_to_number()
is "pid->vnr" in my case and a for() loop in your.

Nevertheless, that's just a guess.

> Suka, right now are you allocating the struct upid separately from the
> struct pid? That alone might slow things down quite a bit. By
> allocating them as one large struct - saving both an alloc at clone, and
> a dereference when looking at pid.upid[0] to get the pid_ns for instance
> - you might get some of this perf back.
>
> (Hmm, taking a quick look, it seems you're allocating the memory as one
> chunk, but then even though the struct upid is just at the end of the
> struct pid, you use a pointer to find the struct upid. That could slow
> things down a bit)

Right now Suka is allocating a struct pid and struct pid_elem as one chunk.
There even exists a kmem cache names pid+1elem :)

> Anyway, Pavel, I'd like to look at some profiling data (when Suka or I
> collects some) and see whether the slowdown is fixable. If it isn't,
> then we should definately look at combining the patchsets.

OK. Please, keep me advised.

> thanks,
> -serge
>
>> 2. The flat pid namespaces (my part)
>> 3. The multilevel pid namespaces (suka's part)
>>
>>> Did you happen to collect any profiling information during your runs?

>> Unfortunately no :(My intention was to prove that hierarchy has
>> performance implications and should be considered carefully.
>>
>>> -- 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: Pid namespaces approaches testing results
Posted by Cedric Le Goater on Fri, 01 Jun 2007 08:47:26 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> Quoting Pavel Emelianov (xemul@openvz.org):
>> Dave Hansen wrote:
>>> On Tue, 2007-05-29 at 15:45 +0400, Pavel Emelianov wrote:
>>>> The detailed results are the following:
>>>> Test name: spawn execl shell ps (sys time)
>>>> 1(no ns) : 579.1 618.3 1623.2 3.052s
>>>> 2(suka's): 570.7 610.8 1600.2 3.107s
>>>> Slowdown : 1.5% 1.3% 1.4% 1.8%
>>>>
>>>> 3(no ns) : 580.6 616.0 1633.8 3.050s
>>>> 4(flat) : 580.8 615.1 1632.2 3.054s
>>>> Slowdown : 0% 0.1% <0.1% 0.1%
>>>> 5(multi) : 576.9 611.0 1618.8 3.065s
>>>> Slowdown : 0.6% 0.8% 0.9% 0.5%
>>> Wow, thanks so much for running those. You're a step ahead of us,
>>> there!
>> Thanks :) Maybe we shall cooperate then and make three series
>> of patches like
>>
>> 1. * The Kconfig options;
>>
>> * The API. I.e. calls like task_pid_nr(), task_session_nr_ns() etc;
>> This part is rather important as I found that some places in kernel
>> where I had to lookup the hash in multilevel model were just pid->vpid
>> dereference in flat model. This is a good optimization.

>>
>> * The changes in the generic code that intruduce a bunch of
>> #ifdef CONFIG_PID_NS
>> ...
>> #else
>> #ifdef CONFIG_PID_NS_FLAT
>> #endif
>> #ifdef CONFIG_PID_NS_MULTILEVEL
>> #endif
>> #endif
>> code in pid.c, sched.c, fork.c etc
>>
>> This patchset will have to make kernel prepared for namespaces injections
>> and (!) not to break normal kernel operation with CONFIG_PID_NS=n.
>
> In principle there's nothing at all wrong with that (imo). But the
> thing is, given the way Suka's patchset is set up, there really isn't
> any reason why it should be slower when using only one or two pid
> namespaces.
>
> Suka, right now are you allocating the struct upid separately from the
> struct pid? That alone might slow things down quite a bit. By
> allocating them as one large struct - saving both an alloc at clone, and
> a dereference when looking at pid.upid[0] to get the pid_ns for instance
> - you might get some of this perf back.
>
> (Hmm, taking a quick look, it seems you're allocating the memory as one
> chunk, but then even though the struct upid is just at the end of the
> struct pid, you use a pointer to find the struct upid. That could slow
> things down a bit)

what about being more agressive and defining :

```
 struct pid
 {
  atomic_t count;
  /* lists of tasks that use this pid */
  struct hlist_head tasks[PIDTYPE_MAX];
  int num_upids;
  struct upid upid_list[CONFIG_MAX_NESTED_PIDNS];
  struct rcu_head rcu;
 };
```

if CONFIG_MAX_NESTED_PIDNS is 1, then pid namespaces are not available.
at 2, the model is flat and at 3, we start nesting them.

it should improve performance as profiling gave higher memory usage
in the current 2.6.21-mm2-pidns3 patchset.

C.

C.

_____

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

---

Subject: Re: Pid namespaces approaches testing results
Posted by Pavel Emelianov on Fri, 01 Jun 2007 09:41:05 GMT
View Forum Message <> Reply to Message

Cedric Le Goater wrote:
> Serge E. Hallyn wrote:
>> Quoting Pavel Emelianov (xemul@openvz.org):
>>> Dave Hansen wrote:
>>>> On Tue, 2007-05-29 at 15:45 +0400, Pavel Emelianov wrote:
>>>>> The detailed results are the following:
>>>>> Test name: spawn execl shell ps (sys time)
>>>>> 1(no ns) : 579.1 618.3 1623.2 3.052s
>>>>> 2(suka's): 570.7 610.8 1600.2 3.107s
>>>>> Slowdown : 1.5% 1.3% 1.4% 1.8%
>>>>>
>>>>> 3(no ns) : 580.6 616.0 1633.8 3.050s
>>>>> 4(flat) : 580.8 615.1 1632.2 3.054s
>>>>> Slowdown : 0% 0.1% <0.1% 0.1%
>>>>> 5(multi) : 576.9 611.0 1618.8 3.065s
>>>>> Slowdown : 0.6% 0.8% 0.9% 0.5%
>>>> Wow, thanks so much for running those. You're a step ahead of us,
>>>> there!
>>> Thanks :) Maybe we shall cooperate then and make three series
>>> of patches like
>>>
>>> 1. * The Kconfig options;
>>>
>>> * The API. I.e. calls like task_pid_nr(), task_session_nr_ns() etc;
>>> This part is rather important as I found that some places in kernel
>>> where I had to lookup the hash in multilevel model were just pid->vpid
>>> dereference in flat model. This is a good optimization.
>>>
>>> * The changes in the generic code that intruduce a bunch of
>>> #ifdef CONFIG_PID_NS
>>> ...
>>> #else
>>> #ifdef CONFIG_PID_NS_FLAT

>>> #endif
>>> #ifdef CONFIG_PID_NS_MULTILEVEL
>>> #endif
>>> #endif
>>> code in pid.c, sched.c, fork.c etc
>>>
>>> This patchset will have to make kernel prepared for namespaces injections
>>> and (!) not to break normal kernel operation with CONFIG_PID_NS=n.
>> In principle there's nothing at all wrong with that (imo). But the
>> thing is, given the way Suka's patchset is set up, there really isn't
>> any reason why it should be slower when using only one or two pid
>> namespaces.
>>
>> Suka, right now are you allocating the struct upid separately from the
>> struct pid? That alone might slow things down quite a bit. By
>> allocating them as one large struct - saving both an alloc at clone, and
>> a dereference when looking at pid.upid[0] to get the pid_ns for instance
>> - you might get some of this perf back.
>>
>> (Hmm, taking a quick look, it seems you're allocating the memory as one
>> chunk, but then even though the struct upid is just at the end of the
>> struct pid, you use a pointer to find the struct upid. That could slow
>> things down a bit)
>
> what about being more agressive and defining :
>
> struct pid
> {
>   atomic_t count;
>   /* lists of tasks that use this pid */
>   struct hlist_head tasks[PIDTYPE_MAX];
>   int num_upids;
>   struct upid upid_list[CONFIG_MAX_NESTED_PIDNS];
>   struct rcu_head rcu;
> };
>
> if CONFIG_MAX_NESTED_PIDNS is 1, then pid namespaces are not available.
> at 2, the model is flat and at 3, we start nesting them.

The flat model has many optimization ways in comparison with the multilevel
one. Like we can cache the pid value on structs and some other.

Moreover having generic level nesting sounds reasonable. Having single level
nesting - too as all the namespace we have are single nested. But having the
4 level nesting sounds strange... Why 4? Why not 5? What if I don't know how
many I will need exactly, but do know that it will be definitely more than 1?

Moreover - I have shown that we can have 1% or less performance on generic

nesting model, why not keep it?

> it should improve performance as profiling gave higher memory usage
> in the current 2.6.21-mm2-pidns3 patchset.
>
> C.
>
>
> C.
>

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

---

Subject: Re: Pid namespaces approaches testing results
Posted by Cedric Le Goater on Tue, 05 Jun 2007 21:16:45 GMT
View Forum Message <> Reply to Message

> The flat model has many optimization ways in comparison with the multilevel
> one. Like we can cache the pid value on structs and some other.
>
> Moreover having generic level nesting sounds reasonable. Having single level
> nesting - too as all the namespace we have are single nested. But having the
> 4 level nesting sounds strange... Why 4? Why not 5? What if I don't know how
> many I will need exactly, but do know that it will be definitely more than 1?
>
> Moreover - I have shown that we can have 1% or less performance on generic
> nesting model, why not keep it?

did you send that patchset ? is it included in the one you sent ?

sorry if i missed something :(

C.

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

---

Subject: Re: Pid namespaces approaches testing results
Posted by Pavel Emelianov on Wed, 06 Jun 2007 07:09:52 GMT
View Forum Message <> Reply to Message

Cedric Le Goater wrote:
>> The flat model has many optimization ways in comparison with the multilevel
>> one. Like we can cache the pid value on structs and some other.
>>
>> Moreover having generic level nesting sounds reasonable. Having single level
>> nesting - too as all the namespace we have are single nested. But having the
>> 4 level nesting sounds strange... Why 4? Why not 5? What if I don't know how
>> many I will need exactly, but do know that it will be definitely more than 1?
>>
>> Moreover - I have shown that we can have 1% or less performance on generic
>> nesting model, why not keep it?
>
> did you send that patchset ? is it included in the one you sent ?

The patchset I sent earlier changed slightly. The tests were performed
on the version I sent. Right now I'm waiting for your results to make
a final decision whether or not to develop the flat model together with
the hierarchical one.

So what are we going to do? The ways we have:
1. Make two models - hierarchical and flat. Maybe we'll see how to merge
   them later;
2. Optimize the hierarchical model to produce no performance hit on the
   first 2 levels (init and VS). I don't see the way to make this
   gracefully, but I maybe this can be solved ... somehow. Anyway, if
   the latest patches from Suka do not produce any noticeable overhead,
   I am OK to go on with them;
3. Make the CONFIG_MAX_NS_DEPTH model. This is likely to be fast in the
   flat case, but I am in doubt whether Andrew will like it :)

> sorry if i missed something :(
>
> C.
>

Thanks,
Pavel

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