Subject: Re: Pid namespaces approaches testing results
Posted by ebiederm on Tue, 29 May 2007 13:00:37 GMT
View Forum Message <> Reply to Message

Pavel Emelianov <xemul@openvz.org> writes:

> 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.

   For small levels of nesting a for loop should actually be faster.

These tests were all taken in the initial pid namespace?
Yes.  You mention that below.

> 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)

If these test accurately measure what the purport to measure
these appear to fair, and useful for discussion.  Although we may have
cache hot vs cache cold effects doing weird things to us.

These results need to be reproduced.

We need to get all of the patches against the same kernel
so we can truly have an apples to apples comparison.

The rough number of pids in the system when the tests are taken needs
to be known.

> 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%

Just for my own amusement.
> 1(no ns) :  579.1  618.3  1623.2  3.052s
> 3(no ns) :  580.6  616.0  1633.8  3.050s
-0.25%  0.3%  -0.65%  0.065%

> 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.

Your results do seem to indicate there is measurable overhead,
although in all cases it is slight.  So if we care about performance
we need to look at things very carefully.

Eric

_____
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 Tue, 29 May 2007 13:31:35 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Pavel Emelianov <xemul@openvz.org> writes:
>
>> 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.
>
>     For small levels of nesting a for loop should actually be faster.

Nope. I thought the same when worked on OpenVZ RSS fractions accounting
and found out that loop and hash lookup are almost the same even for
one-element-length list. I don't know what the problem is exactly but
since then I tend to measure my guesses.

> These tests were all taken in the initial pid namespace?
> Yes.  You mention that below.
>
>> 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)
>
> If these test accurately measure what the purport to measure
> these appear to fair, and useful for discussion.  Although we may have
> cache hot vs cache cold effects doing weird things to us.
>
> These results need to be reproduced.
>
> We need to get all of the patches against the same kernel
> so we can truly have an apples to apples comparison.
>
> The rough number of pids in the system when the tests are taken needs
> to be known.

Sure. cat /proc/slabinfo | grep pid shows ~500 pids/pid+1upids
on each kernel (roughly) before the tests.

>> 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%
>
> Just for my own amusement.

Of course - the base kernels differ.

>> 1(no ns) :  579.1  618.3  1623.2  3.052s
>> 3(no ns) :  580.6  616.0  1633.8  3.050s
>  -0.25%  0.3%  -0.65%  0.065%

Not - but + - the larger the number is the better the result is.

I emphasize - the results of namespaces patches were get against
*the base kernel*. I.e. Suka's patches slow down 2.6.21 by 1.5%.
My patches with flat model slowdown the 2.6.22 kernel by 0%.

I believe that the flat model will slowdown even 2.6.21 kernel for
0%, but Suka's - even 2.6.22 by somewhat similar (about 1-2%).

Yet again: the intention of my measurements are not to prove my
multilevel model is better than Suka's one, but to prove that the
*flat* model is faster than multilevel one and thus must be present
in the kernel as well.

>
>> 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.
>
> Your results do seem to indicate there is measurable overhead,
> although in all cases it is slight.  So if we care about performance
> we need to look at things very carefully.

This is slight for init namespace. In sub-namespace the results
may be worse.

IMHO 1.5% is significant enough. 1.5% here and 0.4% there and 0.6%
over there and we have Xen overhead after all :) And no way to find
out what has happened.

> Eric
>

Thank,
Pavel

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