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