Subject: Re: Using a layered filesystem as private dir?
Posted by Rick van Rein on Thu, 05 Jan 2012 19:12:27 GMT
View Forum Message <> Reply to Message

Hello Scott / others,

Thanks for responding. I have good news for OpenVZ :-)

> /mnt isn't your private directory... it should be /mnt/777/private and of course you need a root directory too.

I will admit that the description in the manual pages about root and private has me confused... is it not true that private is the already-mounted directory that will be used as the root directory, and that this root directory is the place where VZ will mount its own local work-copy?

But I found I also did something else wrong, namely to run "vzctl create" instead of "vzctl start". The latter works fine, I found. Silly me...

> So far as sharing files between containers in a CoW situation, Virtuozzo and Linux-VServer offer those features, but OpenVZ does not. It appears you are trying to engineer your own solution.

Hmm, I've seen what VServer does, which is collecting parts that are the same and then use CoW. This is a de-duplication service (that ZFS could also provide) but it is after-the-fact re-assembly; I would rather build on a designed invariant that certain parts share. As explained I also hope to avoid running repetitious upgrades. The savings in disk space are very good, with 900 MB for a good Debian Squeeze and 35 MB added for Apache and PHP. The buffer cache should share in the disk savings. I don't expect that the buffer cache would be shared between VMs that share blocks with CoW (which isn't really sharing), but I'm not sure.

> I don't know if what you want to do will work or not because I haven't tried it. Nor have I tried ZFS in Linux. I suspect it won't work though... but I do wish you luck.

I have some encouraging figures to show that it does work!

Test run 1, just after reboot, based on empty1+squeeze and empty2+squeeze:

```
: root# time vzctl exec 777 find / > /dev/null 2> /dev/null
: real    0m16.013s
: user    0m0.264s
: sys     0m1.508s
:
```

: root# time vzctl exec 777 find / > /dev/null 2> /dev/null
: real   0m8.083s
: user   0m0.120s
: sys   0m0.540s
:
: root# time vzctl exec 777 find / > /dev/null 2> /dev/null
: real   0m0.764s
: user   0m0.056s
: sys   0m0.252s
:
: root# time vzctl exec 778 find / > /dev/null 2> /dev/null
: real   0m1.722s
: user   0m0.076s
: sys   0m0.596s
:
: root# time vzctl exec 778 find / > /dev/null 2> /dev/null
: real   0m0.602s
: user   0m0.052s
: sys   0m0.260s
:
: root# time vzctl exec 777 find / > /dev/null 2> /dev/null
: real   0m1.159s
: user   0m0.060s
: sys   0m0.300s
:
: root# time vzctl exec 778 find / > /dev/null 2> /dev/null
: real   0m1.152s
: user   0m0.072s
: sys   0m0.296s

The values over 1s also show up after waiting a while, so it appears
to be caused by background processes that empty/reuse some of the
buffer cache, but this effect is the same as on a single machine:
the first find takes long, the next ones are a lot faster. I am
not sure what to think of the second with its half-way timing though.

I think these figures warrant the statement that the buffer cache is
shared between VMs if the disk blocks are. And that is a great saving
and a big advantage compared to the Qemu model used by KVM and Xen!
I suppose the maximum amount of OpenVZ VMs on a hardware machine just
got multiplied by ten or more?

Cheers,
 -Rick