Subject: Re: Which of the virtualization approaches is more suitable for kernel?
Posted by Sam Vilain on Tue, 21 Feb 2006 20:33:41 GMT
View Forum Message <> Reply to Message

Kirill Korotaev wrote:
>>>- fine grained namespaces are actually an obfuscation, since kernel
>>> subsystems are tightly interconnected. e.g. network -> sysctl -> proc,
>>> mqueues -> netlink, ipc -> fs and most often can be used only as a
>>> whole container.
>>I think a lot of _strange_ interconnects there could
>>use some cleanup, and after that the interconenctions
>>would be very small
> Why do you think they are strange!? Is it strange that networking
> exports it's sysctls and statictics via proc?
> Is it strange for you that IPC uses fs?
> It is by _design_.

Great, and this kind of simple design also worked well for the first few iterations of Linux-VServer. However, some people need more flexibility as we are seeing by the wide range of virtualisation schemes being proposed. In the 2.1.x VServer patch the network and (process&IPC) isolation and virtualisation have been kept seperate, and can be managed with seperate utilities. There is also a syscall and utility to manage the existing kernel filesystem namespaces.

Eric's pspace work keeps the PID aspect seperate too, which I never envisioned possible.

I think that if we can keep as much seperation between systems as possible, then we will have a cleaner design. Also it will make life easier for the core team as we can more easily divide up the patches for consideration by the relevant subsystem maintainer.

> - you need to track dependencies between namespaces (e.g. NAT requires
> conntracks, IPC requires FS etc.). this should be handled, otherwise one
> container being able to create nested container will be able to make oops.

This is just normal refcounting. Yes, IPC requires filesystem code, but it doesn't care about the VFS, which is what filesystem namespaces abstract.

> do you have support for it in tools?
> i.e. do you support namespaces somehow? can you create half
> virtualized container?

See the util-vserver package, it comes with chbind and vnamespace which allow creation of 'half-virtualized' containers, though most of the rest of the functionality, such as per-vserver ulimits, disklimits, etc have been shoehorned into the general vx_info structure. As we merge into

the mainstream we can review each of these decisions and decide whether it is an inherantly per-process decision, or more XX_info structures are warranted.

>>this doesn't look very cool to me, as IRQs should
>>be handled in the host context and TCP/IP in the
>>proper network space ...
> this is exactly what it does.
> on IRQ context is switched to host.
> In TCP/IP to context of socket or network device.

That sounds like an interesting innovation, and we can compare our patches in this space once we have some common terms of reference and starting points.

>>the question here is, do we really want to turn it
>>off at all? IMHO the design and implementation
>>should be sufficiently good so that it does neither
>>impose unnecessary overhead nor change the default
>>behaviour ...
> this is the question I want to get from Linus/Andrew.
> I don't believe in low overhead. It starts from virtualization, then
> goes reource management etc.
> These features _definetely_ introduce overhead and increase resource
> consumption. Not big, but why not configurable?

Obviously, our projects have different goals; Linux-VServer has very little performance overhead. Special provisions are made to achieve scalability on SMP and to avoid unnecessary cacheline issues. Once that is sorted out, it's very hard to measure any performance overhead of it, especially when the task_struct->vx_info pointer is null.

However I see nothing wrong with making all code disappear without the kernel config option enabled. I expect that as time goes on, you'd just as soon disable it as you would disable the open() system call. I think that's what Herbert was getting at with his comment.

> Seems, you are just trying to move from the topic. Great.

I always did want to be a Lumberjack!

Sam.