Subject: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by akpm on Sat, 24 Mar 2007 05:42:35 GMT
View Forum Message <> Reply to Message

On Fri, 23 Mar 2007 20:30:00 +0100 Herbert Poetzl <herbert@13thfloor.at> wrote:

>
> Hi Eric!
> Hi Folks!
>
> here is a real world example result from one of my tests
> regarding the benefit of sharing over separate memory
>
> the setup is quite simple, a typical machine used by
> providers all over the world, a dual Pentium D 3.2GHz
> with 4GB of memory and a single 160GB SATA disk running
> a Linux-VServer kernel (2.6.19.7-vs2.2.0-rc18)
>
> the Guest systems used are Mandriva 2007 guests with
> syslog, crond, sshd, apache, postfix and postgresql
> installed and running (all in all 17 processes per guest)
>
> the disk space used by one guests is roughly 148MB
>
> in addition to that, a normal host system is running
> with a few daemons (like sshd, httpd, postfix ...)
>
>
> the first test setup is starting 200 of those guests
> one after the other and measuring the memory usage
> before and after the guest did start, as well as
> recording the time used to start them ...
>
> this is done right after the machine was rebooted, in
> one test with 200 separate guests (i.e. 200 x 148MB)
> and in a second run with 200 unified guests (which
> means roughly 138MB of shared files)

Please define your terms. What is a "separated guest", what is a "unified
guest" and how do they differ?

If a "separated" guest is something in which separate guests will use
distinct physical pages to cache the contents of /etc/passwd (ie: a separate
filesystem per guest) then I don't think that's interesting information,
frankly.

Because nobody (afaik) is proposing that pagecache be duplicated across
instances in this fashion.

We obviously must share pagecache across instances - if we didn't want to do that then we could do something completely dumb such as use xen/kvm/vmware/etc ;)

The issue with pagecache (afaik) is that if we use containers based on physical pages (an approach which is much preferred by myself) then we can get in a situation where a pagecache page is physically in container A, is not actually used by any process in container A, but is being releatedly referenced by processes which are in other containers and hence unjustly consumes resources in container A.   How significant a problem this is likely to be I do not know. And there are perhaps things which we can do about it.