Subject: Re: [Vserver] Re: Container Test Campaign
Posted by Gerrit Huizenga on Thu, 06 Jul 2006 14:12:28 GMT
View Forum Message <> Reply to Message

On Thu, 06 Jul 2006 14:44:23 +0400, Kirill Korotaev wrote:
> Gerrit,
>
> >>>>I assuming you are doing your tests on the same system (i.e. same
> >>>>compiler/libs/whatever else), and you do not change that system over
> >>>>time (i.e. you do not upgrade gcc on it in between the tests).
> >>>
> >>>I hope! :)
> >>
> >>All binaries should be built statically to work the same way inside host/guest or
> >>you need to make sure that you have exactly the same versions of glibc and other
> >>system libraries. At least glibc can affect perforamnce very much :/
> >
> >
> > Ick - no one builds binaries statically in the real world. And,
> > when you build binaries statically, you lose all ability to fix
> > security problems in base libraries by doing an update of that library.
> > Instead, all applications need to be rebuilt.
> >
> > Performance tests should reflect real end user usage - not contrived
> > situations that make a particular solution look better or worse.
> > If glibc can affect performance, that should be demonstrated in the
> > real performance results - it is part of the impact of the solution and
> > may need an additional solution or discussion.
>
> What I tried to say is that performance results done in different
> environments are not comparable so have no much meaning. I don't want us
> to waste our time digging in why one environment is a bif faster or slower than another.
> I hope you don't want too.

I *do* want to understand why one patch set or another is significantly
faster or slower than any other. I think by now everyone realizes that
what goes into mainline will not be some slice of vserver, or OpenVZ
or MetaCluster or Eric's work in progress. It will be the convergance
of the patches that enable all solutions, and those patches will be added
as they are validated as beneficial to all participants *and* beneficial
(or not harmful) to mainline Linux. So, testing of large environments
is good to see where the overall impacts are (btw, people should start
reading up on basic oprofile use by about now ;-) but in the end, each
set of patches for each subsystem will be judged on their own merits.
Those merits include code cleanliness, code maintainainability, code
functionality, performance, testability, etc.

So, you are right that testing which compares roughly similar environments

is good. But those tests will help us identify areas where one solution
or another may have code which provides functionality in some way which
has lower impact.

I do not want to have to dig into those results in great detail if the
difference between two approaches is minor. However, if a particular
area has major impacts to performance, we need to understand how the
approaches differ and why one solution has greater impact than another.
Sometimes it is just a coding issue that can be easily addressed. Sometimes
it will be a design issue indicating that one solution or another has
a design issue which might have been better addressed by another solution.

The fun thing here (well, maybe not for each solution provider) is that
we get to cherry pick the best implementations from each solution, or
create new ones as we go which ultimate allow us to have application
virtualization, containers, or whatever you want to call them.

> Now, to have the same environment there are at least 2 ways:
> - make static binaries (not that good, but easiest way)

This is a case where "easiest" is just plain wrong. If it doesn't match
how people will use their distros and solutions out of the box it has
no real relevence to the code that will get checked in.

> - have exactly the same packages in host/VPS for all test cases.
>
> BTW, I also prefer 2nd way, but it is harder.

Herbert's suggestion here is good - if you can use exactly the same
filesystem for performance comparisons you remove one set of variables.

However, I also believe that if the difference between any two filesystems
or even distro environements doing basic performance tests (e.g.
standardized benchmarks) then there is probably some other problem that
we should be aware of. Most of the standardized benchmarks elimininate
the variance of the underlying system to the best of their ability.
For instance, kernbench carries around a full kernel (quite backlevel)
as the kernel that it builds. The goal is to make sure that the kernel
being built hasn't changed from one version to the next. In this case,
it is also important to use the same compiler since there can be
extensive variation between versions of gcc.

gerrit