Subject: Re: [patch 1/4] Network namespaces: cleanup of dev_base list use
Posted by ebiederm on Mon, 26 Jun 2006 16:26:23 GMT
View Forum Message <> Reply to Message

Andrey Savochkin <saw@swsoft.com> writes:

> Hi Eric,
>
> On Mon, Jun 26, 2006 at 09:13:52AM -0600, Eric W. Biederman wrote:
>> Andrey Savochkin <saw@swsoft.com> writes:
>>
>> > Cleanup of dev_base list use, with the aim to make device list
> per-namespace.
>> > In almost every occasion, use of dev_base variable and dev->next pointer
>> > could be easily replaced by for_each_netdev loop.
>> > A few most complicated places were converted to using
>> > first_netdev()/next_netdev().
>>
>> As a proof of concept patch this is ok.
>>
>> As a real world patch this is much too big, which prevents review.
>> Plus it takes a few actions that are more than replace just
>> iterators through the device list.
>
> dev_base list is historically not the cleanest part of Linux networking.
> I've still spotted a place where the first device in dev_base list is assumed
> to be loopback. In early days we had more, now only one place or two...

I agree. I'm just saying this should be several patches in a patchset
not just one big one.

>> In addition I suspect several if not all of these iterators
>> can be replaced with the an appropriate helper function.
>>
>> The normal structure for a patch like this would be to
>> introduce the new helper function. for_each_netdev.
>> And then to replace all of the users while cc'ing the
>> maintainers of those drivers. With each different
>> driver being a different patch.
>>
>> There is another topic for discussion in this patch as well.
>> How much of the context should be implicit and how much
>> should be explicit.
>>
>> If the changes from netchannels had already been implemented, and all of
>> the network processing was happening in a process context then I would
>> trivially agree that implicit would be the way to go.
>

> Why would we want all network processing happen in a process context?

The basic idea is that an interrupt comes in. A light weigh classifier
looks at the packet and throws it into the appropriate socket packet queue.

Beyond that everything can happen in the socket packet queue in process
context which reduces the number of locks you need, and increases cache
locality.

Van Jacobson's slides showed some impressive system load reductions by
doing that.

The increased locality aids the kind of work we are doing as well,
by meaning we don't have to guess.

It is a big enough problem that I don't think we want to gate on
that development but we need to be ready to take advantage of it when
it happens.

>> However short of always having code always execute in the proper
>> context I'm not comfortable with implicit parameters to functions.
>> Not that this the contents of this patch should address this but the
>> later patches should.
>
> We just have too many layers in networking code, and FIB/routing
> illustrates it well.

I don't follow this comment. How does a lot of layers affect
the choice of implicit or explicit parameters? If you are maintaining
a patch outside the kernel I could see how there could be a win for
touching the least amount of code possible but for merged code that
you only have to go through once I don't see how the number of layers
affects things.

As I recall for most of the FIB/routing code once you have removed
the global variable accesses and introduce namespace checks in
the hash table (because allocating hash tables at runtime isn't sane)
the rest of the code was agnostic about what was going on. So I think
you have touched everything that needs touching. So I don't see
a code size or complexity argument there.

I do agree that we do have a lot of code there.

>> When I went through this, my patchset just added an explicit
>> continue if the devices was not in the appropriate namespace.
>> I actually prefer the multiple list implementation but at
>> the same time I think it is harder to get a clean implementation
>> out of it.

>
> Certainly, dev_base list reorganization is not the crucial point in network
> namespaces. But it has to be done some way or other.
> If people vote for a single list with skipping devices from a wrong
> namespace, it's fine with me, I can re-make this patch.
>
> I personally prefer per-namespace device list since we have too many places
> in the kernel where this list is walked in a linear fashion,
> and with many namespaces this list may become quite long.

I completely agree that cleaning up the list is a good thing to do,
regardless of how we implement things. So that can easily be a pre cursor
to the work. My only practical question is if we can remove the list
walks be calling some of the generic helper functions.

I would say just separate out the list walk cleanup code and submit it.
The important point is that the patch needs to stand on it's as a
cleanup rather than depending on network namespaces as a justification.

Eric
