Subject: Re: [PATCH] [NETNS49] support for per/namespace routing cache cleanup
Posted by Daniel Lezcano on Wed, 17 Oct 2007 14:46:34 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> Denis V. Lunev wrote:
>>> Daniel Lezcano wrote:
>>>> Denis V. Lunev wrote:
>>>>> /proc/sys/net/route/flush should be accessible inside the net
>>>>> namespace.
>>>>> Though, the complete opening of this file will result in a DoS or
>>>>> significant entire host slowdown if a namespace process will
>>>>> continually
>>>>> flush routes.
>>>>>
>>>>> This patch introduces per/namespace route flush facility.
>>>>>
>>>>> Each namespace wanted to flush a cache copies global generation
>>>>> count to
>>>>> itself and starts the timer. The cache is dropped for a specific
>>>>> namespace
>>>>> iff the namespace counter is greater or equal global ones.
>>>>>
>>>>> So, in general, unwanted namespaces do nothing. They hold very old low
>>>>> counter and they are unaffected by the requested cleanup.
>>>>>
>>>>> Signed-of-by: Denis V. Lunev <den@openvz.org>
>>>>
>>>> That's right and that will happen when manipulating ip addresses of
>>>> the network devices too. But I am not confortable with your
>>>> patchset. It touches the routing flush function too hardly and it
>>>> uses current->nsproxy->net_ns.
>>>>
>>>> IMHO we should have two flush functions. One taking a network
>>>> namespace parameter and one without the network namespace parameter.
>>>> The first one is called when a write to
>>>> /proc/sys/net/ipv4/route/flush is done (we must use the network
>>>> namespace of the writer) or when a interface address is changed or
>>>> shutdown|up. The last one is called by the timer, so we have a
>>>> global timer flushing the routing cache for all the namespaces.
>>>
>>> we can't :(The unfortunate thing is that the actual cleanup is
>>> called indirectly and asynchronously. The user _schedule_ the garbage
>>> collector to run NOW and we are moving over a large routing cache.
>>> Really large.
>>>

>>> The idea to iterate over the list of each namespace to flush is bad.
>>> We are in atomic context. The list is protected by the mutex.
>>>
>>> The idea of several timers (per namespace) is also bad. You will
>>> iterate over large cache several times.
>>>
>>> No other acceptable way here for me :(.
>>>
>>> As for "the trigger" - rt_cache_flush, looks like you are right. We
>>> should pass namespace as a parameter. This should be done as a
>>> separate patch.
>>
>> If we change:
>>
>>   rt_cache_flush(struct net *net, int delay);
>>
>> and inside we call rt_cache_flush((unsigned long)net);
>>
>> And then we check in the rt_run_flush function,
>>
>>   struct net *net = (struct net *)dummy;
>>
>>   ...
>>   for (i = rt_hash_mask; i >= 0; i--) {
>>   ...
>>   if (dummy && rth->fl.fl_net != net)
>>   continue
>>   ...
>>   ...
>>
>> So when rt_run_flush is called synchronously, the netns is specified
>> in dummy and only the routes belonging to netns are flushed. Otherwise
>> when it is called by the timer, netns is not set so all routes are
>> flushed.
>>
>
> this does not look good for me. The size of this cache for 4GB host is
> 2*10^6 entries for IPv4 with a 131072 chains. The conventional
> mainstream kernel wants to merge the routing cache cleanup requests from
> the different sources if they are delayed (default).
>
> The main idea for this patch is to protect all other namespaces from the
> current one. This cache is an important resource. You proposal will work
> for the forced synchronous cleanups. Though, there are some requests
> with results in the delayed rt_run_flush via [mod/add]_timer
>
> How should we handle them?

IHMO we should let the timer to remove all routes.

The problem you raised is someone in a namespace can flush routes for another namespaces and mess the network traffic for them.
This case is resolved with the synchronous call to rt_cache_flush and netns parameter.

The second point here is we can have a lot of routes and the timer will flush them all, with or without namespaces.
IMHO I don't think we should handle this case ... for now.
