Subject: Re: Re: [PATCHSET] 2.6.20-lxc8
Posted by Kirill Korotaev on Fri, 23 Mar 2007 09:35:01 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Benjamin Thery <benjamin.thery@bull.net> writes:
>
>
>>My investigations on the increase of cpu load when running netperf inside a
>>container (ie. through etun2<->etun1) is progressing slowly.
>>
>>I'm not sure the cause is fragmentation as we supposed initially.
>>In fact, it seems related to forwarding the packets between the devices.
>>
>>Here is what I've tracked so far:
>>* when we run netperf from the container, oprofile reports that the top
>>"consuming" symbol is: "pskb_expand_head". Next comes
>>"csum_partial_copy_generic". these symbols represents respectively 13.5% and
>>9.7% of the samples.
>>* Without container, these symbols don't show up in the first 20 entries.
>>
>>Who is calling "pskb_expand_head" in this case?
>>
>>Using systemtap, I determined that the call to "pskb_expand_head" comes from the
>>skb_cow() in ip_forward() (l.90 in 2.6.20-rc5-netns).
>>
>>The number of calls to "pskb_expand_head" matches the number of invocations of
>>ip_forward() (268000 calls for a 20 seconds netperf session in my case).
>
>
> Ok. This seems to make sense, and is related to how we have configured the
> network in this case.
>
> It looks like pskb_expand_head is called by skb_cow.
>
> skb_cow has two cases when it calls pskb_expand_head.
> - When there are multiple people who have a copy of the packet
>   (tcpdump and friends)
> - When there isn't enough room for the hard header.
>
> Any chance one of you guys looking into this can instrument up
> ip_foward just before the call to skb_cow and find out which
> reason it is?
>
> A cheap trick to make the overhead go away is probably to setup
> ethernet bridging in this case...
>
> But if we can ensure the ip_foward case does not need to do anything

> more than modify the ttl and update the destination that would
> be good to.
>
> Anyway this does look very solvable.

we have the hack below in ip_forward() to avoid skb_cow(),
Banjamin, can you check whether it helps in your case please?
(NOTE: you will need to replace check for NETIF_F_VENET with something else
 or introduce the same flag on etun device).

```
diff -upr linux-2.6.18-rhel5.orig/net/ipv4/ip_forward.c
linux-2.6.18-rhel5-028stab023/net/ipv4/ip_forward.c
--- linux-2.6.18-rhel5.orig/net/ipv4/ip_forward.c       2006-09-20 07:42:06.000000000 +0400
+++ linux-2.6.18-rhel5-028stab023/net/ipv4/ip_forward.c 2007-03-20 17:22:45.000000000 +0300
@@ -86,6 +86,24 @@ int ip_forward(struct sk_buff *skb)
        if (opt->is_strictroute && rt->rt_dst != rt->rt_gateway)
                goto sr_failed;

+       /*
+        * We try to optimize forwarding of VE packets:
+        * do not decrement TTL (and so save skb_cow)
+        * during forwarding of outgoing pkts from VE.
+        * For incoming pkts we still do ttl decr,
+        * since such skb is not cloned and does not require
+        * actual cow. So, there is at least one place
+        * in pkts path with mandatory ttl decr, that is
+        * sufficient to prevent routing loops.
+        */
+       iph = skb->nh.iph;
+       if (
+#ifdef CONFIG_IP_ROUTE_NAT
+           (rt->rt_flags & RTCF_NAT) == 0 &&     /* no NAT mangling expected */
+#endif                                           /* and */
+           (skb->dev->features & NETIF_F_VENET)) /* src is VENET device */
+               goto no_ttl_decr;
+
        /* We are about to mangle packet. Copy it! */
        if (skb_cow(skb, LL_RESERVED_SPACE(rt->u.dst.dev)+rt->u.dst.header_len))
                goto drop;
@@ -94,6 +112,8 @@ int ip_forward(struct sk_buff *skb)
        /* Decrease ttl after skb cow done */
        ip_decrease_ttl(iph);

+no_ttl_decr:
+
        /*
         *      We now generate an ICMP HOST REDIRECT giving the route
         *      we calculated.
```

@@ -121,3 +141,5 @@ drop:

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers