Subject: Re: Re: [PATCHSET] 2.6.20-lxc8
Posted by dev on Wed, 28 Mar 2007 08:01:41 GMT
View Forum Message <> Reply to Message

Benjamin,

checksumming can be optimized out as well.
We had an experimental patch for OpenVZ venet device, which adds
NETIF_F_LLTX | NETIF_F_HW_CSUM | NETIF_F_SG | NETIF_F_HIGHDMA
features to venet device and avoids additional checksumming where possible
(moving RX/TX checksum calculation to hardware).

So I guess this is doable in future as well.

Thanks,
Kirill

```
> Hi,
>
> Yesterday, I applied a patch similar to Kirill's one that skip
> skb_cow() in ip_forward when the device is a etun, and it does help a lot.
>
> With the patch the cpu load increase is reduced by 50%. Part of the
> problem is "solved".
>
> Here are the figures for netperf:
>
> (Host A -> Host B
>   Host A is running kernel 2.6.20-rc5-netns.i386)
>
>                              Throughput    CPU load
>
> - without container:              719.78     10.45
> - inside a container (no patch)   719.37     21.88
> - inside a container with patch:  728.93     15.41
>
> The CPU load with the ip_forward patch is now "only" 50% higher (10%
> compared to 15%) than the reference case without container.
>
> The throughput is even better (I repeated the test a few times and I
> always got better results from inside the container).
>
> (1) Why skb_cow() performs the copy?
>
> I also added some traces to understand why skb_cow() does copy the
> skb: is it insufficient headroom or that the skb has been cloned
> previously?
> In our case, the condition is always that the "TCP skb" is marked as
```

> cloned.
> It is likely that these skb have been cloned in tcp_skb_transmit().
>
>
> (2) Who consumes the other 5% percent cpu?
>
> With the patch installed oprofile reports that pskb_expand_head()
> (called by skb_cow) has disappeared from the top cpu consumers list.
>
> Now, the remaining symbol that shows unusual activity is
> csum_partial_copy_generic().
> I'd like to find who is the caller, unfortunately, this one is harder
> to track. It is written in assembler and called by "static inline"
> routines and Systemtap doesn't like that. :(
>
>
> So, that was the current status.
> I'm continuing my investigations.
>
> Regards,
> Benjamin
>
> Eric W. Biederman wrote:
>
>>Kirill Korotaev <dev@openvz.org> writes:
>>
>>
>>>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).
>>
>>Ugh. The thing is skb_cow should be free. It only has a cost when the skb
>>is too small or there is a second copy of the skb. I don't there is a technical
>>reason for either of those to be the case when we are going over ethernet.
>>
>>And since the hardware header needs to change as well your hack is actually broken
>>if the incoming network interface is not ethernet.
>>
>>So while I can see this hack for testing I'd much rather see if we can actually
>>fix this one cleanly.
>>
>>Unless you understand what is triggering the skb_cow to actually perform
>>the copy.
>>
>>Eric
>>
>>

```
>>>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