Case study: Network

David Chang
dchang@suse.com

bsc#1005184 - Kernel update slows down partner application making it 7-10X slower

- SLE11 SP3
- Reported by: Juston Mortenson
- Assigned To: Benjamin Poirier
- wasL3:46549

Synopsis

- Partner has an application that is used in a container inside a container
- This application passes recon image data via the network loopback device using a sender and receiver.
- Partner’s developer says "From profiling I can see that the sender process appears to spend a lot of time pending on mutexes. So the issue is either **networking or futex** related”

Synopsis

- Update kernel version from 3.0.101-0.47.52 to 3.0.101-0.47.79
- Tested with different kernel version
  - 3.0.101-0.47.86 slow
  - 3.0.101-0.47.52 fast
  - 3.0.101-0.47.55 slow

Finding

- Application, more detail about it
  - Application uses DDS (from RTI) to transmit data from a producer to a consumer. This uses RTPS (Realtime Publish Subscribe) protocol, which is on top of UDP, to transport the data
  - Issue happens when the producer and consumer reside on the same host using the localhost interface, the overall transaction runs slowly

- Kernel
  - Problem was introduced between **3.0.101-0.47.52.1 and 3.0.101-0.47.55.1**
  - Kernel regression ?

Changelog of the kernel

$ kernel-source> git log --oneline --no-merges rpm-3.0.101-0.47.52..rpm-3.0.101-0.47.55 | wc -l

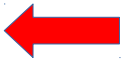
236

- Network

  fa7563b5b3fa net: llc: use correct size for sysctl timeout entries (bsc#919007, CVE-2015-2041).

  D88fe8be9cd9 ipv4: Missing sk_nulls_node_init() in ping_unhash() (bsc#929525, CVE-2015-3636).

  07fbbcb57188 net: rds: use correct size for max unacked packets and bytes (bsc#919018 CVE-2015-2042)

  f032bebc2275 net: relax rcvbuf limits (bug#923344)  git bisect

- Other

  e3ca8a9e4684 deal with deadlock in d_walk fix (bnc#929148, bnc#929283)

0fd7bac6b615 net: relax rcvbuf limits (1)

```
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -637,12 +637,14 @@ static inline void __sk_add_backlog(struct sock *sk,
struct sk_buff *skb)

 /* Take into account size of receive queue and backlog queue
+ * Do not take into account this skb truesize,
+ * to allow even a single big packet to come.
  */
 static inline bool sk_rcvqueues_full(const struct sock *sk, const struct sk_buff
*skb)
 {
        unsigned int qsize = sk->sk_backlog.len + atomic_read(&sk
                                        ->sk_rmem_alloc);

-       return qsize + skb->truesize > sk->sk_rcvbuf;
+       return qsize > sk->sk_rcvbuf;
```

0fd7bac6b615 net: relax rcvbuf limits (2)

```
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -288,11 +288,7 @@ int sock_queue_rcv_skb(struct sock *sk,
struct sk_buff *skb)
        unsigned long flags;
        struct sk_buff_head *list = &sk->sk_receive_queue;

-       /* Cast sk->rcvbuf to unsigned... It's pointless, but reduces
-          number of warnings when compiling with -W --ANK
-       */
-       if (atomic_read(&sk->sk_rmem_alloc) + skb->truesize >=
-           (unsigned)sk->sk_rcvbuf) {
+       if (atomic_read(&sk->sk_rmem_alloc) >= sk->sk_rcvbuf) {
```

0fd7bac6b615 net: relax rcvbuf limits (3)

commit 0fd7bac6b6157eed6cf0cb86a1e88ba29e57c033
Author: Eric Dumazet <eric.dumazet@gmail.com>
Date: Wed Dec 21 07:11:44 2011 +0000

net: relax rcvbuf limits

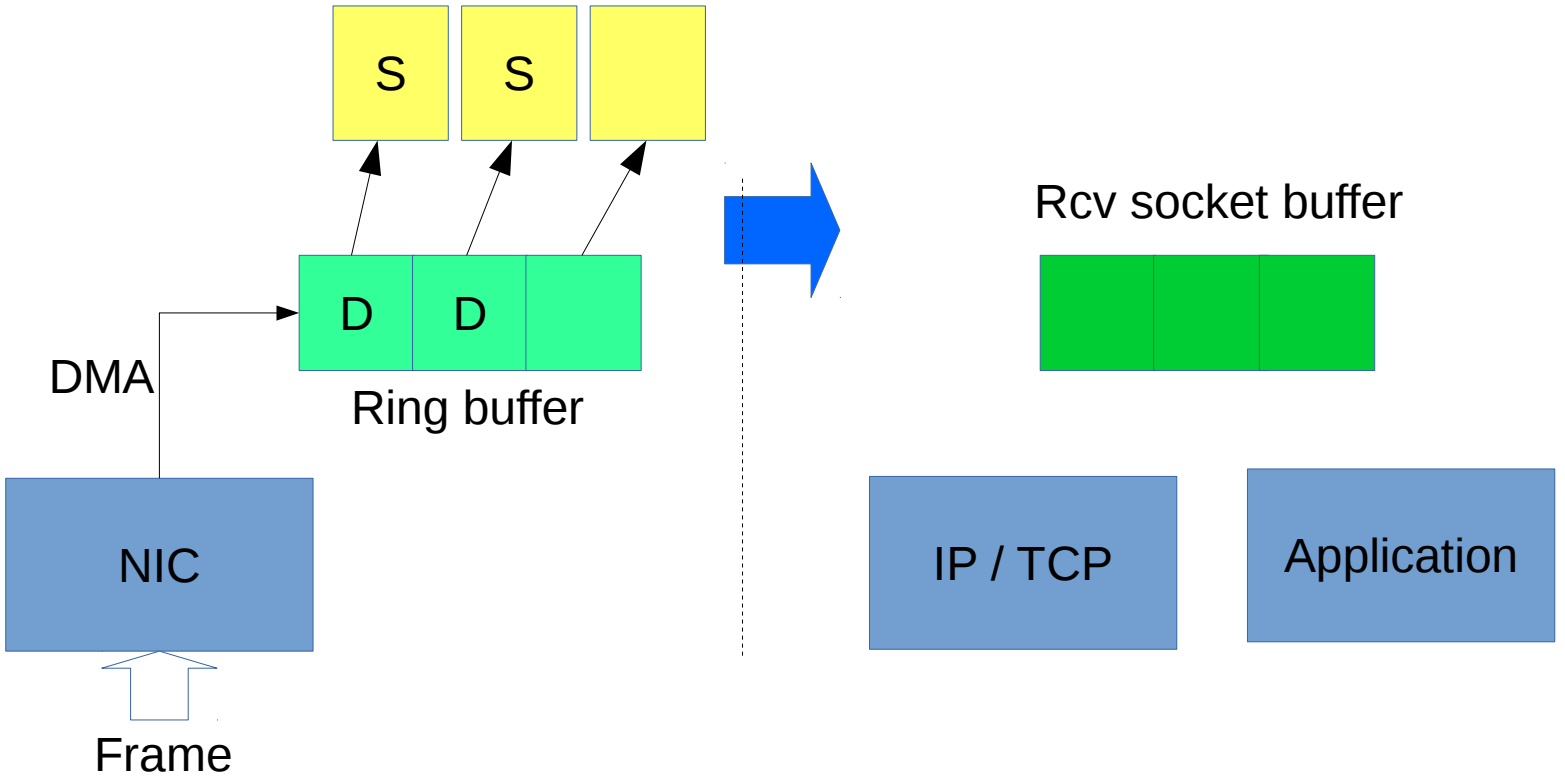
skb->truesize might be big even for a small packet.

Its even bigger after commit 87fb4b7b533 (net: more accurate skb truesize) and big MTU.

We should allow queueing at least one packet per receiver, even with a low RCVBUF setting.

Receive flow

- Packet is transferred from NIC to ring buffer
- Packet is transferred from ring buffer to a receive socket buffer



Added trace events

- 3847ce32aea9 core: add tracepoints for queueing skb to rcvbuf (v3.1-rc1)
  - **sock_rcvqueue_full**
  - **sock_exceed_buf_limit**
- **sock_rcvqueue_almost_full**
  - Add it to sk_rcvqueues_full()

Additional trace event (1)

```
TRACE_EVENT(sock_rcvqueue_almost_full,

    TP_PROTO(const struct sock *sk, const struct sk_buff *skb, void
*location),

    TP_ARGS(sk, skb, location),

    TP_STRUCT__entry(
        __field(void *, location)
        __field(const void *, sk)
        __field(int, sk_backlog_len)
        __field(int, rmem_alloc)
        __field(unsigned int, truesize)
        __field(int, sk_rcvbuf)
    ),
```

Additonal trace event (2)

```
    TP_fast_assign(
        __entry->location = location;
        __entry->sk = sk;
        __entry->sk_backlog_len = sk->sk_backlog.len;
        __entry->rmem_alloc = atomic_read(&sk->sk_rmem_alloc);
        __entry->truesize   = skb->truesize;
        __entry->sk_rcvbuf  = sk->sk_rcvbuf;
    ),

    TP_printk("location=%p sk=%p backlog.len=%d sk_rmem_alloc=%d truesize=%u
sk_rcvbuf=%d",
        __entry->location,
        __entry->sk,
        __entry->sk_backlog_len,
        __entry->rmem_alloc,
        __entry->truesize,
        __entry->sk_rcvbuf)
);
```

Trace log

- The trace look like:

  <...>-43081 [017] 6078.310577: sock_rcvqueue_almost_full: location=ffffffff813f86e8 sk=ffff881e1acd0240 family=2 **b**acklog.len=0 **sk_rmem_alloc=213728 truesize=65880 sk_rcvbuf=262142**

  <...>-43081 [017] 6078.310605: sock_rcvqueue_full: location=ffffffff813fb7e0 **rmem_alloc=279608 truesize=65880 sk_rcvbuf=262142**

- return qsize + skb->truesize > sk->sk_rcvbuf;

+ return qsize > sk->sk_rcvbuf;

Socket receive buffer size

- rmem_default (/proc/sys/net/core/rmem_default)
  - controls the default size of receive buffers used by sockets
  - sysctl -w net.core.rmem_default=N
- SO_RCVBUF
  - Control the maximum socket receive buffer in bytes
  - To set this by calling setsockopt()

Conclusion

- The analysis shows that tuning of the net.core.rmem_max is what is needed to get the application to work correctly after the kernel update
- A kernel regression? Not really.
- Kernel event tracing is useful.

Events for networking

- udp:udp_fail_queue_rcv_skb
  - __udp_queue_rcv_skb() ← udp_queue_rcv_skb()
  - net/ipv4/udp.c

- sock:sock_exceed_buf_limit
  - __sk_mem_raise_allocated() ← __sk_mem_schedule()
  - net/core/sock.c
  - added to __sk_mem_raise_allocated (old: __sk_mem_schedule) and it records limitations of memory for sockets and current usage

- **sock:sock_rcvqueue_full**
  - __sock_queue_rcv_skb ← sock_queue_rcv_skb()
  - net/core/sock.c
  - added to sock_queue_rcv_skb(). It records rcvbuf size and its usage

Events for networking

- napi:napi_poll
  - napi_busy_loop() ←
    - ep_busy_loop() [fs/eventpoll.c]
    - sk_busy_loop() [include/net/busy_poll.h]
  - napi_poll() ←
    - napi_busy_loop()
    - net_rx_action()
  - net/core/dev.c
  - poll_one_napi() ← poll_napi()
  - net/core/netpoll.c

- net:netif_rx_ni_entry
  - netif_rx_ni() ←drivers
  - net/core/dev.c

Events for networking

- net:netif_rx_entry
  - netif_rx() ← drivers
  - net/core/dev.c
  - receives a packet from a device driver and queues it for the upper layer to process
- net:netif_receive_skb_entry
  - netif_receive_skb() ← drivers
  - net/core/dev.c
  - main receive data processing function, process receive buffer from network
- net:napi_gro_receive_entry
  - napi_gro_receive() ← drivers
  - net/core/dev.c

Events for networking

- net:napi_gro_frags_entry
  - napi_gro_frags() ← drivers
  - net/core/dev.c
- net:netif_rx
  - netif_rx_internal() ←
    - dev_forward_skb()
    - netif_rx()
    - netif_rx_ni()
  - net/core/dev.c
- net:netif_receive_skb
  - __netif_receive_skb_core() ← __netif_receive_skb()
  - net/core/dev.c

Events for networking

- net:net_dev_queue
  - __dev_queue_xmit() ←
    - dev_queue_xmit()
    - dev_queue_xmit_accel()
  - Queue a buffer for transmission to a network device
  - net/core/dev.c

- net:net_dev_xmit
  - xmit_one() ← dev_hard_start_xmit()
  - net/core/dev.c

- net:net_dev_start_xmit
  - xmit_one() ← dev_hard_start_xmit()
  - net/core/dev.c

Events for networking

- skb:skb_copy_datagram_iovec
  - skb_copy_datagram_iter() ←
    - skb_copy_datagram_msg()
    - skb_copy_datagram_iter() [net/core/datagram.c]
    - tap/tun drivers
  - Copy a datagram to an iovec iterator
  - net/core/datagram.c

- skb:consume_skb
  - net_tx_action() [net/core/dev.c]← No user
  - consume_skb() ← drivers
  - consume_stateless_skb() ←skb_consume_udp()
  - napi_consume_skb() ← drivers tx irq handler
  - net/core/skbuff.c

- skb:kfree_skb
  - net_tx_action() ← No user
  - kfree_skb() ← drivers


SUSE®
We adapt. You succeed.