recvmmsg
Batch datagram processing

#netconf, September, 2009

Arnaldo Carvalho de Melo
acme@redhat.com

Why?

. Finantial Institutions
. Trading
. Lots of small packets
. Multicast
. Reducing cost of net stack entry/exit

How?

. Receive multiple datagrams per syscall
. recvmsg extension
. sendmsg later

First step - upper layer

. recvmmsg
. struct msghdr iovec
. timeout for whole operation
. repeteadely calls underlying recvmsg
. Cost of syscalls reduced
. security_recvmsg only on the first datagram
. cost of SELinux reduced

API

Returns number of datagrams

Parameters

- int fd,
- struct mmsghdr *mmsg,
- unsigned vlen,
- unsigned flags,
- struct timespec *timeout

Second step - lower layer

. unlocked_recvmsg
. lock_sock at batch start/release_sock at batch end
. fallback to recvmsg
. UDP did it only at packet freeing (mem accounting)
. Now its for the whole batch
. other locks to remove
. TCPfy this brother

Nir Tzachar’s numbers

. Publisher/Mirror/Consumer
. Publisher + Consumer on the same machine
. Better latency & Troughput for small packets
. Bigger packets exhibit similar latency but better throughput
. 1 Gbit/s, at 10 Gbit/s it should show even better improvements
. 100-byte packets in batches of 30 reduces latency from 750us to 470us

recvmsg profile

```
# Overhead Command   Shared Object          Symbol                 # Samples: 1169433
    6.18%  recvmmsg  [kernel]               [k] _spin_lock_irqsave
           |--32.82%-- remove_wait_queue
           |          free_poll_entry
           |          poll_freewait
           |          sys_poll
           |          system_call_fastpath
           |          __GI___poll
           |--26.47%-- add_wait_queue
           |          __pollwait                                  <SNIP>
           |--10.75%-- __skb_recv_datagram
           |          udp_recvmsg                                    <SNIP>
    4.91%  recvmmsg  [kernel]               [k] _spin_lock_bh
           |--56.70%-- lock_sock_nested
           |          lock_sock
           |          udp_recvmsg
           |          sock_recvmsg
           |          sys_recvmsg
           |          system_call_fastpath
           |          __libc_start_main
           |--28.50%-- udp_poll
           |          sock_poll                                   <SNIP>
           |--12.42%-- release_sock
           |          udp_recvmsg                                    <SNIP>
```

First step - Perf numbers for recvmmsg

```
# Overhead   Command          Shared Object  Symbol                  # Samples: 582228
   11.10%  recvmmsg  [kernel]               [k] _spin_lock_bh
           |--78.12%-- lock_sock
           |          udp_recvmsg
           |          sock_common_recvmsg
           |          __sock_recvmsg_nosec
           |          |--98.52%-- sock_recvmsg_nosec
           |          |          __sys_recvmsg
           |          |          __sys_recvmmsg
           |          |          sys_recvmmsg
           |          |          system_call_fastpath                    <SNIP>
           |           --1.48%-- __sock_recvmsg
           |                     sock_recvmsg                          <SNIP>
           |--19.99%-- release_sock
           |          udp_recvmsg
           |          sock_common_recvmsg
           |          __sock_recvmsg_nosec
           |          |--97.55%-- sock_recvmsg_nosec                      <SNIP>
           |           --2.45%-- __sock_recvmsg                         <SNIP>
           |--0.74%-- udp_poll
           |          sock_poll                                  <SNIP>
    4.44%  recvmmsg  [kernel]               [k] sub_preempt_count
           |--30.81%-- _local_bh_enable_ip
           |--26.73%-- _spin_unlock_irqrestore
           |          |--49.12%-- __skb_recv_datagram
           |          |--42.52%-- skb_queue_tail
           |          |          sock_queue_rcv_skb
           |          |          __udp_queue_rcv_skb
           |          |          sk_backlog_rcv
           |          |          release_sock
           |          |          udp_recvmsg                          <SNIP>
    4.15%  recvmmsg  [kernel]               [k] copy_user_generic_string       <SNIP>
```

Second step - Perf numbers

```
# Overhead   Command          Shared Object  Symbol                    # Samples: 761074
    6.54%  recvmmsg  [kernel]              [k] skb_set_owner_r
           |--99.43%-- sock_queue_rcv_skb
           |          __udp_queue_rcv_skb
           |          sk_backlog_rcv
           |          release_sock
           |          __sys_recvmmsg                                  <SNIP>
    5.88%  recvmmsg  [kernel]              [k] _spin_lock_irqsave
           |--47.58%-- skb_queue_tail
           |          sock_queue_rcv_skb
           |          __udp_queue_rcv_skb
           |          sk_backlog_rcv
           |          release_sock
           |          __sys_recvmmsg
           |--41.85%-- __skb_recv_datagram
           |          __udp_recvmsg
           |          udp_unlocked_recvmsg
           |          sock_common_unlocked_recvmsg
           |          __sock_unlocked_recvmsg_nosec
           |          |--98.41%-- sock_unlocked_recvmsg_nosec
           |          |          __sys_recvmsg
           |          |          __sys_recvmmsg                       <SNIP>
           |           --1.59%-- sock_unlocked_recvmsg
           |                     __sys_recvmsg                      <SNIP>
    5.28%  recvmmsg  [kernel]              [k] copy_user_generic_string       <SNIP>
    5.21%  recvmmsg  [kernel]              [k] release_sock                 <SNIP>
```

Now the counterpart: sendmmsg

. Also iovec
. Send same payload to multiple destinations
. Not repeating payload, 0 sized means use last iovec entry