o2iblnd Queue Depth, Credit Management and Concurrent Sends

Credit Management Logic

- When a connection is established on the passive or active side `connibc_credits` is set to the negotiated queue depth
  - For the passive side this happens in `kiblnd_passive_connect()`
  - For the active side this happens in `kiblnd_check_connreply()`
- When posting a transmit, `kiblnd_post_tx_locked()` is passed a credit value.
  - The credit value depends on the message. It's either 0 or 1.
  - `IBLND_MSG_PUT_NAK, IBLND_MSG_PUT_ACK, IBLND_MSG_PUT_DONE, IBLND_MSG_GET_DONE, IBLND_MSG_NOOP` (for v2 protocol) require no credits
  - All other messages consume 1 credit.
  - If the message consumes a credit then subtract that from con->ibc_credits;
  - If we fail to post the message we return the credit to the connection
  - When packing the message `msgibm_credits` is set to `connibc_outstanding_credits` (`connibc_outstanding_credits` is described below)
- When handling a receive in `kiblnd_handle_rx()`, depending on the message we'll either increment `connibc_outstanding_credits` or `connibc_reserved_credits` by 1 after we call `ib_post_recv()` to receive buffers.
  - The `ibc_reserved_credits` is used to transfer messaged for the `ibc_tx_queue_rsrved` queue to to the `ibc_tx_queue` for sending.
- When it comes time to post a transmit, then as shown above, the `ibc_outstanding_credits` is assigned to `msgibm_credits` and sent to the peer

The general effect of that algorithm

1. When posting a transmit the connection credits is reduced by 1
2. When handling a message if the transmit is returning any credits then we add them back to the connection credits and attempt to send outstanding messages, which takes us to (1)
3. When we post buffers to receive we increment the `ibc_outstanding_credits`. This is passed in the next transmit to the peer. That takes us back to (2)

The underlying assumption in the connection management algorithm is that both sides are exchanging messages. If there is a change in the call flow where one side simply sends events with the other side not responding using IMMEDIATE messages, the initiating side will run out of credits and will be stuck since none of the credits are being returned.

That's why we have the NOOP message. The intent is to monitor the outstanding credits on the connection and once they pass the peer_credits_hiw mark, then the NOOP message eagerly returns the credits to the other side of the connection to allow it to continue sending.

Queue Depth Negotiation

Queue depth is negotiated as follows:

- active creates its qp and sends it's queue depth to the passive
- passive creates its end of the qp and then sends back its own queue depth which should be <= of the active's
- active receive's the passive's queue depth and sets that to the connection queue depth and credits. Now both ends have the same queue depth and starting credits.

017d328fa832697533e4e032fe9a9213ea105320 LU-10213 lnd: calculate qp max_send_wrs properly leverages this algorithm by decreasing the active's queue depth when attempting to create the qp, then sending the adjusted queue depth to the passive. The passive creates its connection structure and reduces the queue depth if necessary, then sends it back to the active, which uses that as the connection queue depth and credits.

There are some MLX5 limitation when calculating the max_send_wr which relies on the negotiatied Queue Depth.

From: ~~LU-7124~~ - MLX5: Limit hit in cap.max_send_wr RESOLVED

```
Here is the reply I got from a Mellanox engineer:
Hi,
I sent in the past an explanation to this list and I am going to repeat it.
The number reported for max_qp_wr is the maximum value the HCA
supports. But it is not guaranteed that this maximum is supported for
any configuration of a QP. For example, the number of send SGEs and the
transport service can affect this max value.

From the spec:
11.2.1.2 QUERY HCA
Description:
Returns the attributes for the specified HCA.
The maximum values defined in this section are guaranteed not-to-exceed
values. It is possible for an implementation to allocate some HCA
resources from the same space. In that case, the maximum values returned
are not guaranteed for all of those resources simultaneously

Mlx5 supported devices work as described above. Mlx4 supported
devices has some flexibility allowing it to user larger work queues so
this is why you can define 16K WRs for mlx4 and for mlx5 you can do only
 8K (in your specific case).
```

Snippet from a discussion with MLX engineer

```
max_qp_wr;              / Maximum number of outstanding WR on any work queue /

I checked max_qp_wr on the cards I have:
ConnecX-3, ConnectX-3 Pro (mlx4): 16351
Connect-IB, ConnectX-4, ConnectX-5 (mlx5): 32768
Amir, Looks like max_send_wr and max_recv_wr are limited by ib_device_attr.max_qp_wr
But I see a lot of other checks at QP creating in drivers/infiniband/hw/mlx5/qp.c:create_kernel_qp() and drivers
/infiniband/hw/mlx5/qp.c:calc_sq_size(). I aslo see the following note "There may be RDMA devices that for
specific transport types may support less outstanding Work Requests than the maximum reported value"
```

Concurrent Sends

Concurrent sends were intended to limit the maximum number of in-flight transfers for the entire system. However, we were multiplying the max_send_wr with concurrent sends which implied that it's per connection, which is not true.

It is better to remove concurrent sends tunable completely as that will simplify the code and instead rely on the queue depth to limit the in-flight transfers per connection.

The jury is still up on this change. It needs to be tested in the filed to see if it'll have a negative impact on performance.

Patches

- https://review.whamcloud.com/28279 LU-9810 lnd: use less CQ entries for each connection
- https://review.whamcloud.com/29995 LU-10129 lnd: rework map_on_demand behavior
- https://review.whamcloud.com/30309 LU-10129 lnd: set device capabilities
- https://review.whamcloud.com/30310 LU-10213 lnd: calculate qp max_send_wrs properly
- https://review.whamcloud.com/30311 LU-9943 lnd: correct WR fast reg accounting
- https://review.whamcloud.com/30312 LU-10291 lnd: remove concurrent_sends tunable