Network Layers

Analogous to

TCP
IP
Ethernet

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>IB Transport</td>
<td></td>
</tr>
<tr>
<td>IB Network</td>
<td></td>
</tr>
<tr>
<td>IB Data Link</td>
<td></td>
</tr>
<tr>
<td>IB Physical</td>
<td></td>
</tr>
</tbody>
</table>

Host Channel Adapter (HCA)
Addressing

• Data Link Layer: **LID** (Local IDentifier)
  • Similar to **MAC Address** in Ethernet.

• Network Layer: **GID** (Global IDentifier)
  • Similar to **IP Address** in IP.

• Transport Layer: **QP Number**
  • Similar to **TCP port** in TCP
Infiniband Transport

• Message Based (unlike TCP for example)

• A variety of message types:
  • Send, RDMA Write, RDMA Read, RDMA Atomics

• Important objects:
  • MR, QP, WQE, CQ, CQE.
MR (Memory Region)

- Similar to virtual memory in CPU / GPU
- MKey (Memory Key) defines a virtual address space
- Memory Region = MKey, Virtual Address, Length
  - Mapped to physical memory via HCA’s MMU.
Memory Registration

- `mr = ibv_reg_mr(…, void *ptr, int length, …)`

- `ptr` is in CPU’s virtual memory.

- IB device driver pins the memory, translates `ptr` to physical address, and maps this address to HCA’s virtual memory.

- Now memory is accessible from the HCA (local and remote) via `mr.key` and `mr.address`. 
**QP**

- Queue Pair (Send Queue and Receive Queue)
- Identified by a QP Number.
- Roughly similar to a socket.
- A Local QP is Connected to a Remote QP (in RC Transport service).
WQE

- \( QP = SQ + RQ \)

- In SQ: Each WQE is a request to send a message.
  - Different info needed for different operations.

- In RQ: Each WQE is a list of buffers (SGL List).
CQ

- Completion Queue
- Contains notifications about completed WQEs
  - Called CQEs (CQ Elements)
- Every Queue is connected to a CQ (e.g. SQ and RQ might be connected to different CQs).
- Rule: 1 WQE $\iff$ 1 CQE
Scatter/Gather List

Memory

Key: 0x123
Address: 0xabcd
Length: 0x1000

S/G Entry (SGE)

Key: 0x126
Address: 0x100
Length: 0x20010

S/G Entry (SGE)

Key: 0x236
Address: 0x7890
Length: 0x100

S/G Entry (SGE)
Send/Recv Operation

• Receiver (Also called responder), posts a RECV WQE to its RQ. (just a list of buffers).

• Sender (Also called requester), posts a SEND WQE to its SQ.
Send/Recv Operation

Memory

QP

Send WQE

Opcode = SEND

Scatter/Gather List

Recv WQE

Scatter/Gather List
Send/Recv Operation

Memory

QP

Send WQE

 Opcode = SEND

Scatter/Gather List

Recv WQE

Scatter/Gather List
Send/Recv Operation

Memory

QP

SQ

Send WQE

RQ

Opcode = SEND

Scatter/Gather List

Recv WQE

Scatter/Gather List

Memory
Send/Recv Operation

Memory

CQ

QP

SQ

RQ

Send WQE

 Opcode = SEND

Scatter/Gather List

Ack
Send/Recv Operation

Memory

CQ
SQ
RQ

CQE
QP

CQE
QP

CQ
RQ
SQ

Memory
RDMA Write Operation

Memory

QP

SQ

Write WQE

Opcode = Write

Scatter/Gather List

Remote Key: 0x123
Remote Address: 0x0

MKey = 0x123
RDMA Write Operation

Memory

QP

SQ

Write WQE

RQ

Opcode = Write

Scatter/Gather List

Remote Key: 0x123
Remote Address: 0x0

SQ

RQ

QP

Memory

MKey = 0x123
RDMA Write Operation

- **Memory**
  - **QP**
    - **SQ**
    - **Write WQE**
    - **RQ**
  - **Write WQE**
    - Opcode = Write
    - Scatter/Gather List
    - Remote Key: 0x123
    - Remote Address: 0x0

- **Remote Key**: 0x123
RDMA Write Operation

- Memory
- QP
  - SQ
  - Write WQE
  - RQ
- QP
  - RQ
  - SQ

Write WQE

- Opcode = Write
- Scatter/Gather List
- Remote Key: 0x123
- Remote Address: 0x0

Ack
RDMA Write Operation

Memory

CQ

CQE

SQ

RQ

QP

MKey = 0x123

CQE

SQ

RQ

QP

Memory
RDMA Write with Immediate

• Same as RDMA Write. Except..

• Generates a CQE in the receiver.

• And thus requires a Receive WQE in the receiver.

  • Why?

• RDMA Write WQE has another field: Immediate. This is a value which shows in the receiver’s CQE.
RDMA Read

- Send (Requester) WQE Contains …
- Receive (Responder) WQE Contains …
Atomics

• Flow similar to read.

• Except operation is read-modify-write.

• RDMA Supports:
  • Compare & Swap
  • Fetch & Add

• Always returns old value. Why?
How to post a WQE

• `ibv_post_send(...)` for requester WQEs
• `ibv_post_recv(...)` for responder WQEs
• WQE is called “`struct ibv_wr`” in verbs.
Posting a send WQE (i.e. ibv_post_send() under the hood)

- Each WQ (Work Queue: SQ / RQ) in the QP is managed as a ring buffer in the host’s memory.
- Has a producer index and a consumer index.
- Who is the producer and who is the consumer?
Posting a send WQE (i.e. ibv_post_send() under the hood)

• Software writes a WQE to the WQ ring buffer.

• HCA does not know yet.

• Software writes the new producer index to a special memory area associated with the QP. Called the doorbell record.

• If HCA just came to look it would see that there is new work …
Posting a send WQE
(i.e. ibv_post_send() under the hood)

• How to tell the HCA to come and look in the Work Queue?
Posting a send WQE (i.e. ibv_post_send() under the hood)

• Poll all the queues?

• slow ...

• could be very very slow ...

• Infiniband supports up to 16M QPs per HCA.

  • Not to talk about SR-IOV: Each VF can have 16M QPs.
Posting a send WQE (i.e. ibv_post_send() under the hood)

- So.. HCA Cannot poll the QPs. We need to ring a bell and tell it to look.

- This operation is actually called: Ringing the doorbell.

  - Just a write to a certain location in the HCA’s BAR. (Different place for different QPs of course).
Posting a send WQE (i.e. ibv_post_send() under the hood)

• So..

• Write WQE to SQ

• Update producer index in Doorbell Record.

• Ring the doorbell.
Posting a send WQE (i.e. `ibv_post_send()` under the hood)

- So..

- Write WQE to SQ

- Update producer index in Doorbell Record.

- Ring the doorbell.

- 3 Writes. What happens if they get reordered?
  - How to prevent that?
Application’s flow

cq, qp =

    initialize_cqs_and_qps(); /* a lot of boilerplate here */

ptr = malloc(size);

mr = ibv_reg_mr(ptr);

ibv_wr wr = {.type = WRITE, .sg.key = mr.key, ....,
             rkey = 0x1234, raddr, 12345}

ibv_post_send(qp, wr);

do {
    n_cqes = ibv_poll_cq(cq, &cqe)
}
while (n_cqes == 0);
rsockets

• Programmers are familiar with sockets interface.

• Infiniband Verbs is very flexible, but many times more difficult to programmers.

• rsockets is a user-space library on top of infiniband verbs which provides sockets interface. Used RDMA Write under the hood.

• Can even run applications built for regular sockets via LD_PRELOAD.