OPENFABRICS ALLIANCE

12th ANNUAL WORKSHOP 2016

EFFECTIVE CODING WITH OFED APIS FOR PERFORMANCE

Adhiraj Joshi, Principal Software Engineer

Veritas Technologies

Apr 6th, 2016

VERITAS

AGENDA

- **Our RDMA journey**
- **Acronyms**
- **Send side optimization**
  - S1: True RDMA – Parallelism
  - S2: Header coalescing
  - S3: Batch Work Requests
  - S4: Avoid fragmentation (for large sends)
- **Receive side optimization**
  - R1: Interrupt spread
  - R2: Optimal interrupt handling
- **Further exploration**
  - Multiple QPs per link?
  - RDMA to remote disk?
  - Etc.
- **Conclusion**

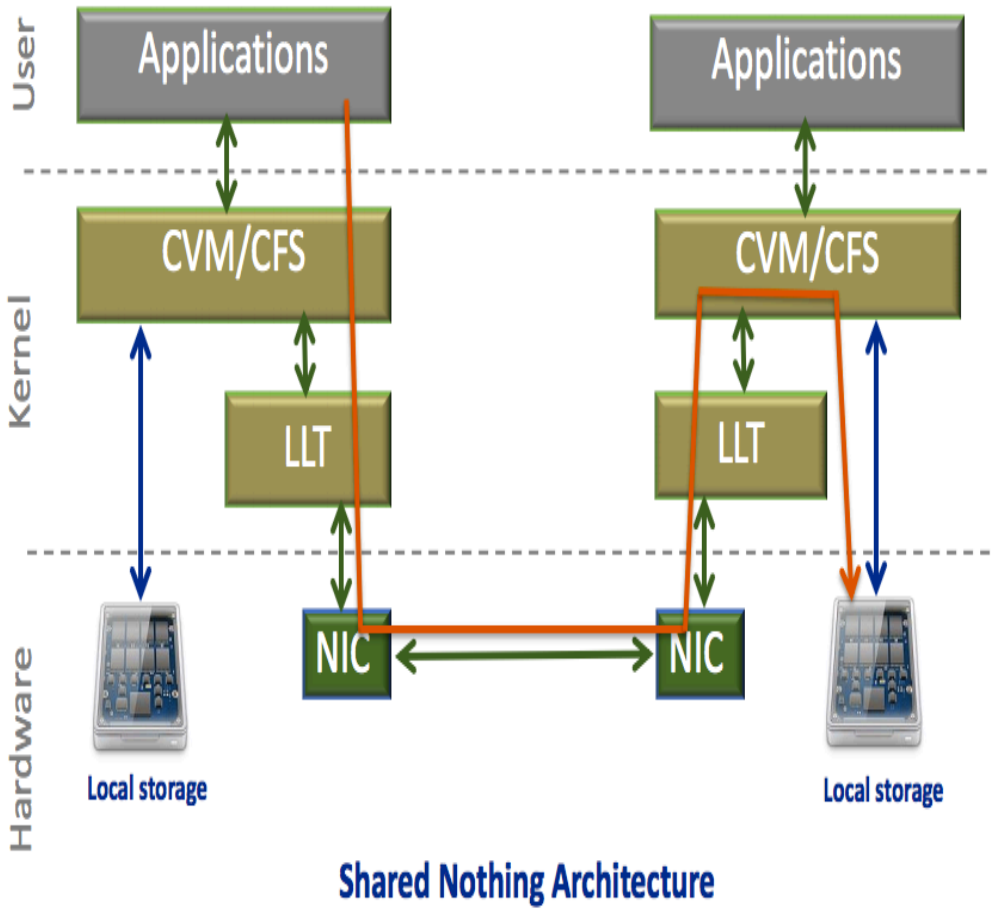
ACRONYMS

Acronym	Full form
LLT	Low Latency Transport (Proprietary network protocol)
CVM	Clustered Volume Manager
CFS	Clustered File System
FSS	Flexible Shared Storage

OUR RDMA JOURNEY

- **Started with basic RDMA functionality**

- **OFA paper 2014: https://www.openfabrics.org/images/eventpresos/workshops2014/DevWorkshop/presos/Tuesday/pdf/01_RDMA_Symantec.pdf**



OUR RDMA JOURNEY (CONTD..)

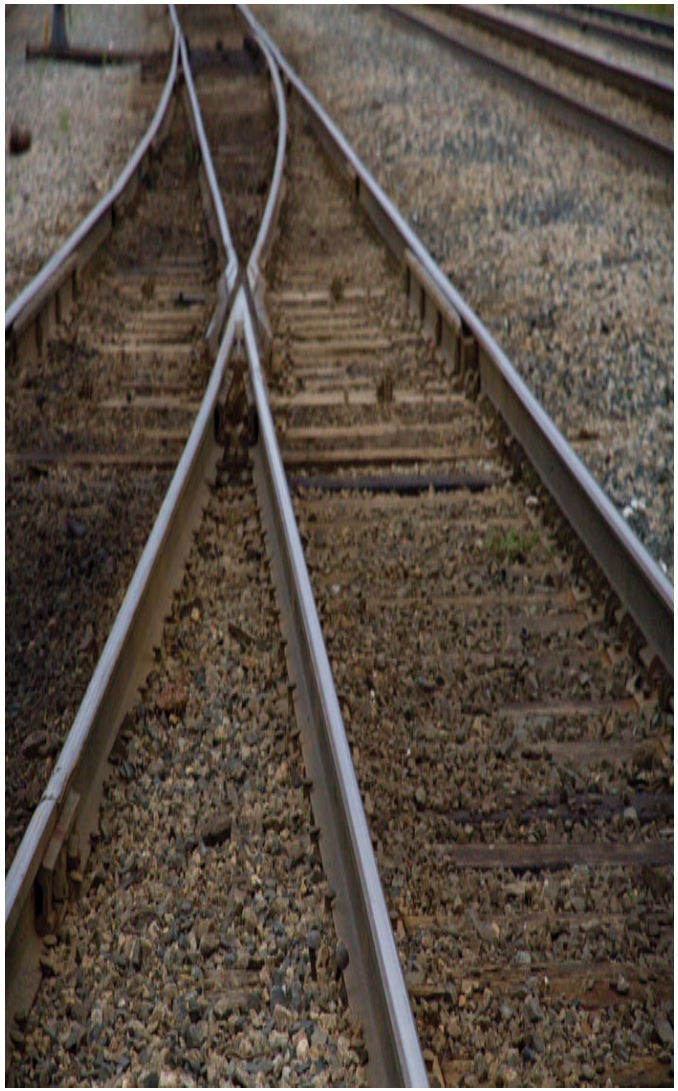
Performance work..

- Network performance at the core of Shared Nothing Architecture
- Need to use OFED APIs optimally
- RDMA journey takes us to different performance improvements
- Achieved big performance improvement

SEND SIDE OPTIMIZATIONS

S1. PARALLELISM (TRUE RDMA)

- **Earlier design**
  - Packets processed in application thread contexts
  - Then queued for RDMA transfer
  - Single thread does the RDMA writes
  - Parallelism lost
  - Three contexts contend on the queue
    - Application threads' context
    - RDMA write context
    - RDMA ACK context
  - Performance bottleneck



S1. CONTD..

- **New design:**
  - Earlier design required queuing to handle broken links
  - New design eliminates queues by leveraging IB_WC_WR_FLUSH error provided by OFED
    - Contains reference to our packet.
- **The Work Completion event:**
  - struct ib_wc {
    u64 wr_id; <— This gives reference to our packet
    enum ib_wc_status status; (FLUSH err)

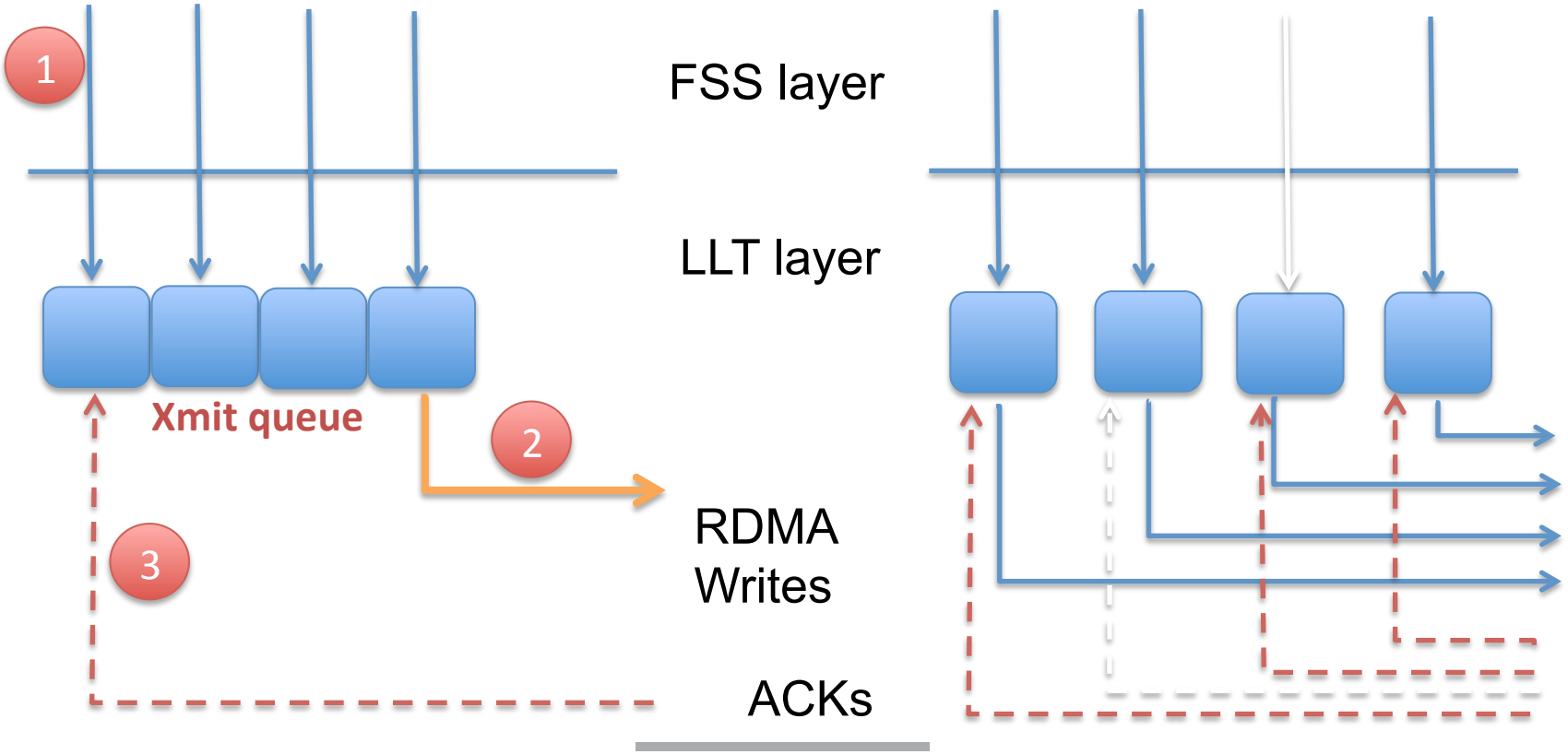
S1. CONTD..

Earlier design

- Queuing involved
- Single LLT xmit thread
- Locking contention

New design

- No Queuing!
- Write with application threads
- No locking!



S1. CONTD..

Performance numbers..

Packet size	Throughput improvement
8K	5%
16K	12%
32K	20%
64K	28%
128K	28%

The receive buffers are of 8K each in all the cases above

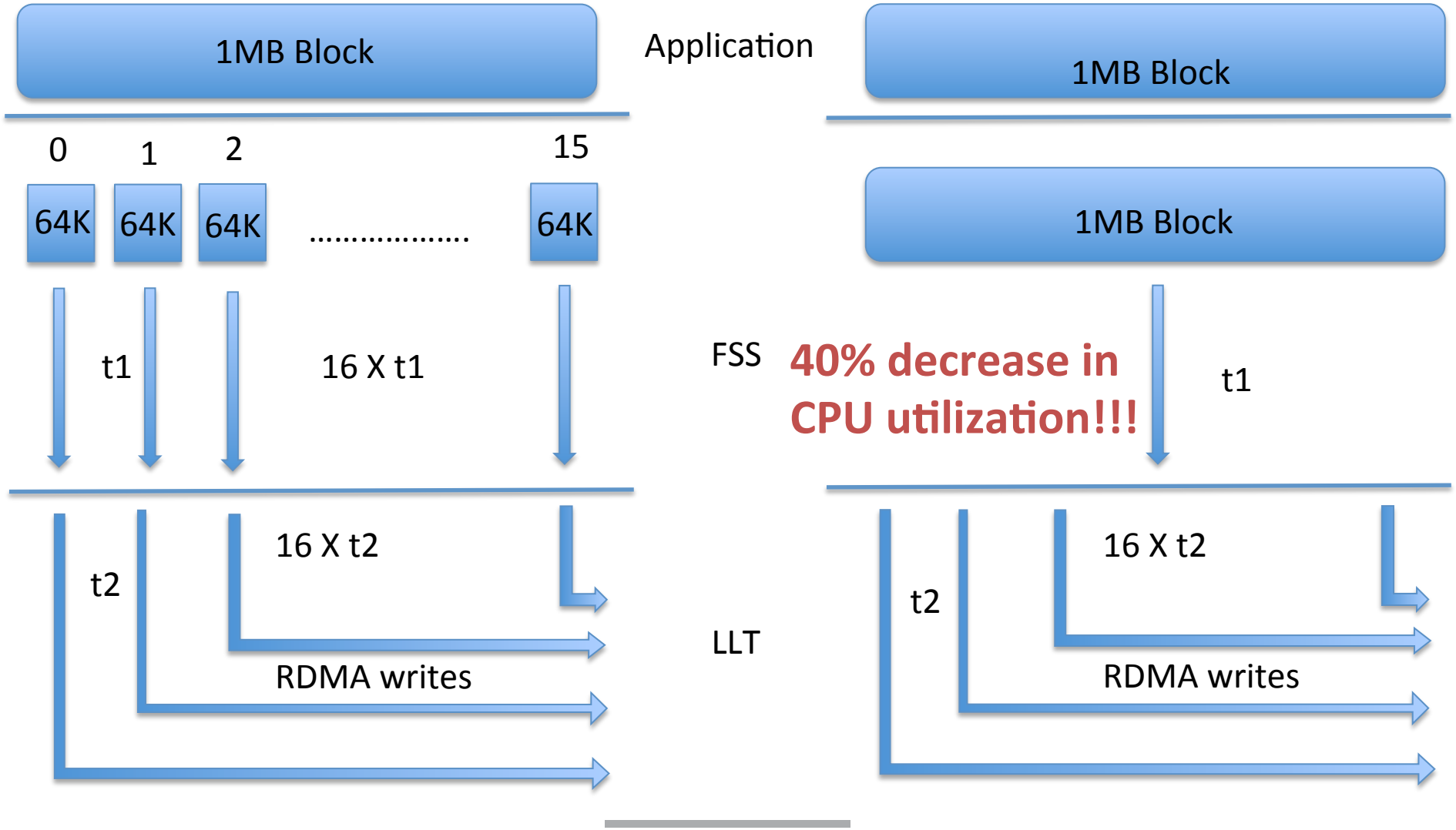
S2. COALESCE HEADER AND DATA

- Application message consists of hdr and data
- Earlier design involved two different RDMA writes
- Makes sense to combine header and data in one RDMA write
- Can reduce network traffic and increase throughput.
- Around 25% improvement over our network module (8K sized IO) based on our network tools

S3. BATCH MULTIPLE WORK REQUESTS

- ib_post_send takes just a few nanosecs
- In case of a highly multithreaded application, this time goes as high as 8 microseconds!
- OFED allows us to batch different work requests
- call ib_post_send() only once.
- Ib_post_send() on failure returns failed work requests
- No performance gain seen, but a nice to have feature

S4. AVOID FRAGMENTATION (FOR LARGE SENDS)



RECEIVE SIDE OPTIMIZATIONS

R1. COMPLETION QUEUE CREATION

- **Problem: Interrupts only on one IRQ line!**
  - # cat /proc/interrupts | grep mlx

```
141:          0          0          0          0          0          0          0          0          0
0          0          0
0          0          0          0          0          0          0          0     917314          0
0          0          0
0          0          0          0          0          0          0  IR-PCI-MSI-edge      mlx4-
ib-1-2@PCI Bus 0000:0a

143:          0          0          0          0          0          0          0          0          0
0          0          0
0          0          0          0          0          0          0          0          0          0          0
0
0          0          0          0          0          0  8          0  IR-PCI-MSI-edge      mlx4-
ib-1-4@PCI Bus 0000:0a
```

R1. COMPLETION QUEUE CREATION

- Use different comp_vector values in the ib_create_cq() for different links
- Else all interrupts come on the same IRQ line
- We lose receive side parallelism
- Maximum usable comp_vectors is given by the num_comp_vectors filed of the ib_device structure (device pointer in struct rdma_cm_id)
- comp_vector = linkno % r_cm_id->device->num_comp_vectors;

R1. CONTD..

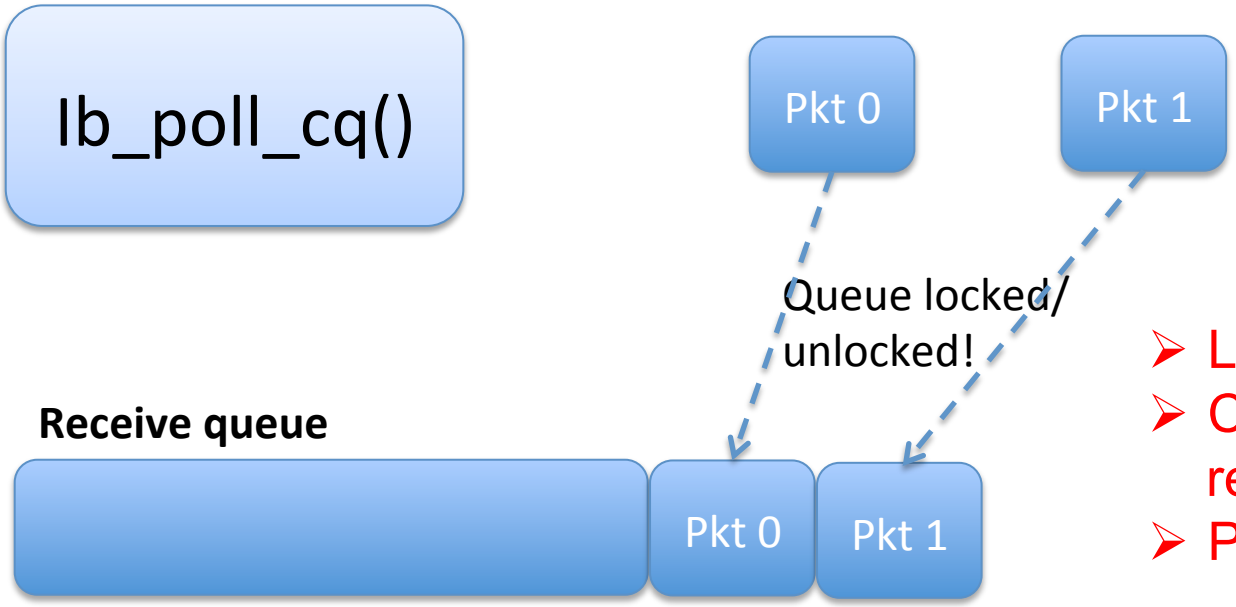
- **Interrupts evenly spread!**
- **27% improvement over performance improvement in S1 (i.e. total S1 + R1 gives 55% improvement)**

  - # cat /proc/interrupts | grep mlx

```
141:          0          0          0          0          0          0          0          0          0          0
0          0
0          0          0          0          0          0          0          0     432254          0          0
0          0
0          0          0          0          0          0          0  IR-PCI-MSI-edge      mlx4-
ib-1-2@PCI Bus 0000:0a

143:          0          0          0          0    0          0          0          0          0          0          0
0
0          0          0          0          0          0     461960          0    0          0          0          0
0
0          0          0          0          0    0  8          0  IR-PCI-MSI-edge      mlx4-
ib-1-4@PCI Bus 0000:0a
```

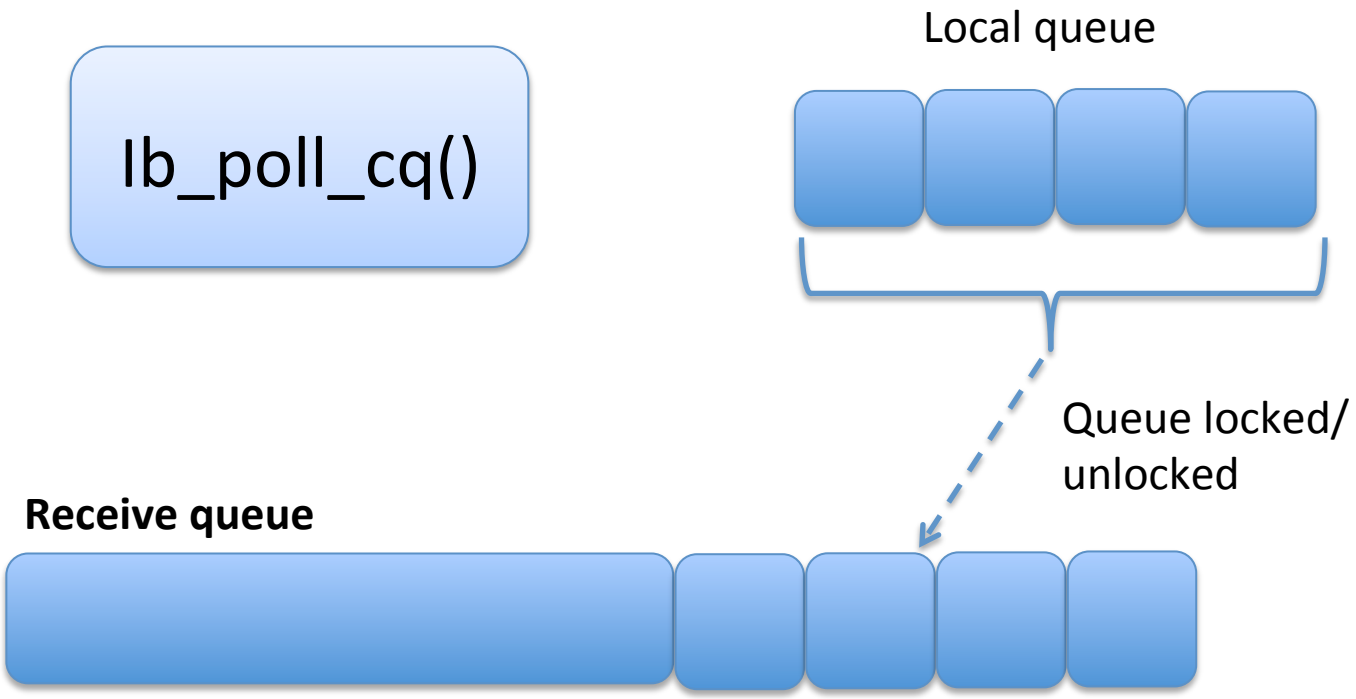
R2. OPTIMAL INTERRUPT HANDLING



- Locking for every packet!
- Contention on the receive queue
- Performance bottleneck

R2. CONTD..

Optimized receive side intr processing..

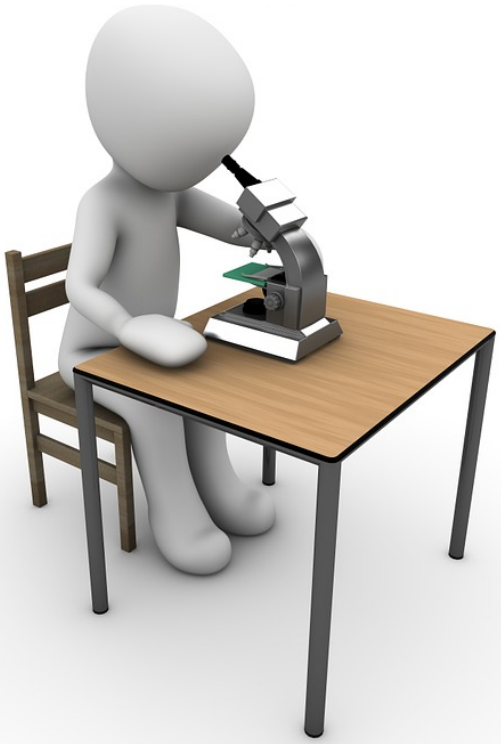


- **Take lock only once for many packets!**
- **We need this to enable perf improvement in S1 and R1**

FURTHER EXPLORATIONS

FURTHER EXPLORATION..

- **Using multiple QPs for one connection**
  - Performance runs with ib_write_bw utility with multiple QPs gives small throughput benefit (5%)
  - Need to explore whether big improvement with multiple QPs is actually expected
- **SCSI RDMA protocol**
  - Write directly to remote disk
- **Batch ACKs**
- **Combine multiple send requests**
- **Find and remove further bottlenecks**



CONCLUSION

- Our RDMA journey started with basic functionality
- Effective use of OFED API helped in huge perf benefit
- Journey continues..

OPENFABRICS
ALLIANCE

12th ANNUAL WORKSHOP 2016

THANK YOU!

Adhiraj Joshi, Principal Software Engineer

adhiraj.joshi1@veritas.com

Veritas Technologies

VERITAS™