RDMA over Ethernet in the WAN

Ezra Kissel
University of Delaware

April 23rd 2012
Internet2 Spring Member Meeting
High-performance data movement

• Increasing demand on networks
  – Better connectivity, more data
  – Petabyte-scale computing

• Increasing WAN capacity
  – 10G -> 100G+
  – Virtual circuits, SDN, smarter networks

• End of CPU frequency scaling
  – 10Gb/s TCP fully taxes a core
  – New approaches needed

• Known issues surrounding existing transport protocols over LFNs
Remote Direct Memory Access (RDMA)

- Supported by Infiniband and more recently over Ethernet (RoCE)
- Connects virtual buffers which may be located in different physical address space...even across a network
- No kernel buffer copies or OS context switching to transfer data
- Virtual-to-physical address translation done by the NIC
RDMA operations

- Infiniband supports reliable (RC) and unreliable (UC) RDMA connections
  - We are interested in the one-sided channel semantics

- RDMA READ/WRITE : Application “pins” memory, exchanges pointers and keys, and then “posts” the desired operations
  - Also known as Get/Put

- All completed RDMA events are written to a “completion queue”
  - Queue is polled for events
  - All operations are asynchronous

- Supported through IB verbs and connection manager libraries
  - `ibverbs`, `rdmacm`
  - OpenFabrics Enterprise Distribution (OFED)
RDMA benchmarks/tools

• rdma_lat
  – One-way delay in microseconds
  – rdma_lat -c -n 100 10.10.10.1

• rdma_bw
  – Simple uni/bi-directional bandwidth test using RDMA WRITE
  – rdma_bw -d mlx0 -c -s 1048576 10.10.10.1

• ib_read_bw/ib_write_bw/ib_send_bw
  – Similar to rdma_bw with some additional options

• ib_bench
  – Even more features: multiple QPs, rate limiting, rx/tx queue depth, different RDMA operations, RC/UC, etc.

• xfer_test (Our native RDMA client)
  – Compare TCP/SDP/RDMA
  – Supports file transfers
  – Socket splicing
Filling the pipe when sending real data

• Must pipeline operations to keep the network saturated
  – Decouple reading and writing to/from buffer regions
  – Fill the transmit queue

• Configurable ring buffer keeps data “in-flight”
  – “ready” partitions can be posted as soon as possible
  – Further partitioned based on demand

4/23/2012
Internet2 Spring Member Meeting
Example testbed environment

- 6-node, 10G connected testbed
  - Mellanox/Myricom adapters, native RoCE support

- Netem and PSpacer enable latency and bottleneck emulation

- Spirent XGEM network impairment device
  - Validates netem
RoCE over WAN performance

- **clean path**
- **.01% loss**
- **.01% reordering**
SC11 SRS Demo Network
SC11 SRS Demo Tests (78ms path)

Native RDMA application

```
# ./xfer_test -c 192.168.140.2 -t 180 -i 2 -a 4 -o 27 -r -x sc-viz/5006
6741: | port=18515 | ib_port=1 | tx_depth=16 | sl=0 | duplex=0 | cma=1 |
Created SLAB buffer with SIZE: 134217728 PARTITIONS: 4
raddr: 0x2aaaaf6ed010, laddr: 0x2aaaaf873010, size: 33554432
raddr: 0x2aaab16ee010, laddr: 0x2aaab1874010, size: 33554432
raddr: 0x2aaab36ef010, laddr: 0x2aaab3875010, size: 33554432
raddr: 0x2aaab56f0010, laddr: 0x2aaab5876010, size: 33554432
Metadata exchange complete
[0.0-2.0 sec]  2.32 GB  9.26 Gb/s
[2.0-4.0 sec]  2.42 GB  9.66 Gb/s
[4.0-6.0 sec]  2.45 GB  9.79 Gb/s

GridFTP using Phoebus RDMA Gateway

```
# globus-url-copy -t 60 -vb -p 1 -dcstack
phoebus:`phoebus_path=192.168.140.1/5006#192.168.140.2/5006`,
xsp:`xsp_hop=sc-viz/5006;xsp_net_path=OPENFLOW`
Source: ftp://192.168.140.1:2811/dev/
Dest:   ftp://192.168.140.2:2811/dev/
    zero -> null

66428338176 bytes  1151.84 MB/sec avg  1164.20 MB/sec inst  [ 9.3 Gb/s ]
Session Layer Burst Switching (SLaBS)

- Apply burst-switching concepts at session-enabled gateways (Phoebus)
  - Send relatively large PDUs ("slabs") versus small layer-3, layer-4 PDUs common today
  - Take advantage of large, fast buffers within the network

- Optimize transmission over dedicated network resources
  - Minimize total transfer time of bulk data flows
  - Latency of individual segments is of less concern

- Use more efficient protocols and reduce overhead
Buffering and bursting with SLaBS

- **Observation**: better utilization can be achieved with N elements sending at maximum rate for 1/Nth of time slot versus N elements competing for 1/Nth of bandwidth

- Contention can lead to loss, so avoid it!

- Buffering required in order to multiplex incoming data into right-sized bursts

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>RKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x70df4</td>
<td>13143242</td>
</tr>
<tr>
<td>0x65a3a</td>
<td>13412431</td>
</tr>
<tr>
<td>0x39c4d</td>
<td>24255433</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SESS ID</th>
<th>OFFSET</th>
<th>LENGTH</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>0</td>
<td>4</td>
<td>0x6BEF</td>
</tr>
<tr>
<td>s2</td>
<td>4</td>
<td>9</td>
<td>0x863B</td>
</tr>
<tr>
<td>s3</td>
<td>13</td>
<td>3</td>
<td>0xD329</td>
</tr>
</tbody>
</table>
Bulk Asynchronous Get

- SLAB buffer information is communicated over XSP control channel
  - Pipeline Get operations up to transmit queue depth
  - Wait for SLAB acknowledgements

- Consumer performs RDMA READ to copy remote buffers
Phoebus-SLaBS performance

GridFTP transfers over dedicated 10G path, increasing WAN latency, 4ms LAN RTT and .001% edge loss
SLaBS over bottlenecks

4 parallel streams, 5G WAN bottleneck with increasing latency
Overhead reduction

Linux `perf` profiling of SLaBS buffer implementation. Comparing TCP to *splicing* and RDMA transport.

Comparing CPU load of SLaBS gateway using TCP, TCP-*splicing*, and RDMA.
Summary

• Short story: RoCE can work well over the WAN
  – Reduced overhead, consistent performance
  – Requires clean paths, buffer management

• A number of tools to experiment with

• RDMA for Phoebus-SLaBS
  – Transparently brings RoCE advantages to existing applications

• Prototype Phoebus deployment on Internet2 backbone
  – Looking for users!
  – https://damsl.cis.udel.edu/phoebus