Implementing TCP Sockets over RDMA

Patrick MacArthur <pio3@cs.unh.edu>
Robert D. Russell <rdr@cs.unh.edu>

Department of Computer Science
University of New Hampshire
Durham, NH 03824-3591, USA

2nd Annual 2014 InfiniBand User Group Workshop
April 3, 2014 2:30pm

Acknowledgements

The authors would like to thank the University of New Hampshire InterOperability Laboratory for the use of their RDMA cluster for the development, maintenance, and testing of UNH EXS. We would also like to thank the UNH-IOL and Ixia for the use of an Anue network emulator for performance testing.

This material is based upon work supported by the National Science Foundation under Grant No. OCI-1127228 and under the National Science Foundation Graduate Research Fellowship Program under award number DGE-0913620.

Outline

1 Background

2 RSockets

3 UNH EXS

4 Performance Evaluation

5 Conclusions

6 References

Differences Between RDMA and TCP Sockets

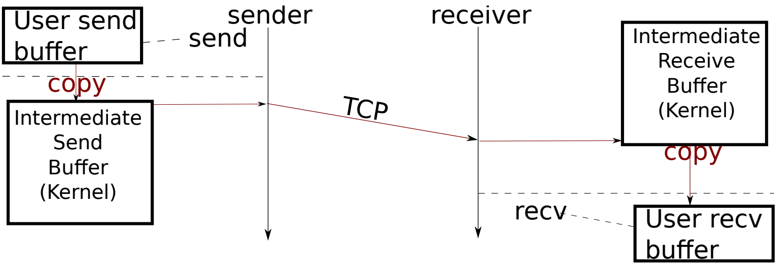
RDMA

- "Kernel bypass": data transfers with no OS involvement
- "Zero-copy": Direct virtual memory to virtual memory transfers
- Message-oriented
- Asynchronous programming interface

TCP Sockets

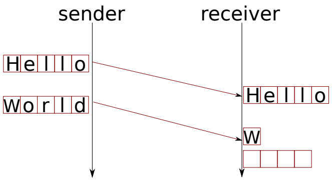
- Kernel involvement in all data transfers
- Buffered in kernel-space on both sides of connection
- Byte-stream oriented protocol
- Synchronous programming interface

TCP Sockets Data Transfer

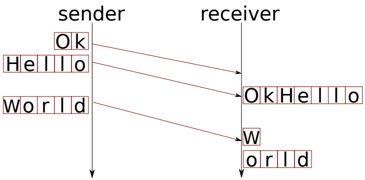


Message vs. Byte Stream Semantics

Message Transfer (RDMA, UDP, SOCK_SEQPACKET)



Byte Stream Transfer (TCP/IP)



Issue: O_NONBLOCK is *not* asynchronous

- Try-and-fail
  - **recv**: if no data in buffer, fail immediately with EAGAIN
  - **send**: if buffer is full, fail immediately with EAGAIN
- POSIX poll/select notify when an operation can start, **not** when operations complete
  - **recv**: poll/select returns when data in buffer
  - **send**: poll/select returns when empty space in buffer
- This is **incompatible** with RDMA semantics
  - RDMA: send or recv **queued**, call returns immediately, proceeds in background

Issue: Implementing "Zero-copy"

- Memory to be used for RDMA must (currently) be **registered**
- Existing sockets programs do not register memory to be used in I/O operations
  - May use any malloc'd/stack variable
  - May be freed at any time
  - Sockets programmers assume memory can be reused as soon as send() returns
- Not respecting adapter's natural alignment can cause severe performance degradation, especially on FDR adapters

Prior Implementations of Sockets over RDMA

- Sockets Direct Protocol (SDP) (defined by InfiniBand specification [InfiniBand 2011])
    - BCopy (buffering on both sides)
    - ZCopy (zero-copy, send() blocks) [Goldenberg 2005]
    - AZ-SDP (asynchronous, zero-copy, segfault handler) [Balaji 2006]
- uStream (asynchronous but not zero-copy) [Lin 2009]

Current Implementations of Sockets over RDMA

- SMC-R (100% compatibility with TCP/IP and sockets)
- rsockets (high-performance sockets replacement) [Hefty 2012]
- UNH EXS (extended sockets) [ISC 2005, Russell 2009, MacArthur 2014]

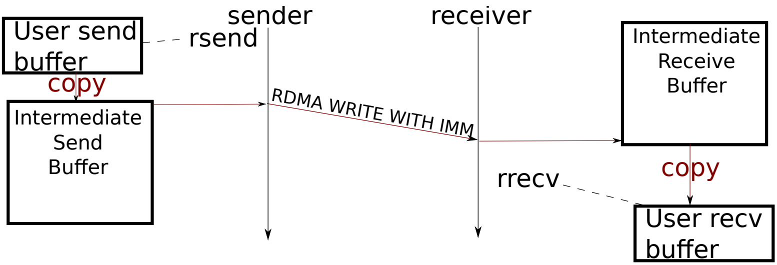
Outline

1. Background
2. RSockets
3. UNH EXS
4. Performance Evaluation
5. Conclusions
6. References

RSockets

- Goal: compatibility with sockets, high performance
- Built on RDMA, so kernel bypass for data transfer path
- Buffer copies on both sides of connection
- Supports SOCK_STREAM (TCP-like) and SOCK_DGRAM (UDP-like) modes
- API is currently synchronous only

RSocket Data Transfer with rsend/rrecv



all in user space

"Zero-copy" with rsockets

- Can perform zero-copy using riomap and riowrite
- **riomap** maps a virtual memory region to an **offset**
- **riowrite** directly transfers data to iomap'd buffer identified by offset

Example

```
/*************** at receiver ****************/
off_t target_offset = riomap(fd, target_buf, len, PROT_WRITE, -1);
rsend(fd, &target_offset, sizeof(target_offset), 0);
rrecv(fd, empty, sizeof(empty), MSG_WAITALL);

/*************** at sender ****************/
off_t target_offset;
rrecv(fd, &target_offset, sizeof(target_offset), MSG_WAITALL);
/* write big buffer to server */
riowrite(fd, local_buf, length, target_offset, 0);
/* notify recipient of completion */
rsend(fd, &empty, sizeof(empty), 0);
```

Outline

1 Background

2 RSockets

3 UNH EXS

4 Performance Evaluation

5 Conclusions

6 References

UNH EXS (Extended Sockets)

- Based on ES-API (Extended Sockets API) published by the Open Group [ISC 2005]
- Extensions to sockets API to provide asynchronous, zero-copy transfers
  - Memory registration (exs_mregister(), exs_mderegister())
  - Event queues for completion of asynchronous events (exs_qcreate(), exs_qdequeue(), exs_qdelete())
  - Asynchronous operations (exs_send(), exs_recv(), exs_accept(), exs_connect())
- UNH EXS supports SOCK_SEQPACKET (reliable message-oriented) and SOCK_STREAM (reliable stream-oriented) modes

UNH EXS Programming

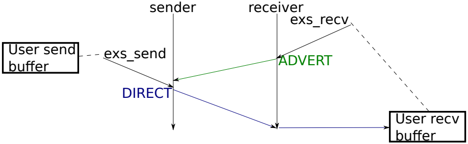
Example

Example asynchronous send operation

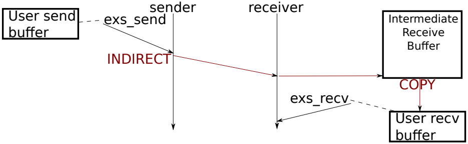
```
exs_mhandle_t mh = exs_mregister(buf, bufsize, EXS_ACCESS_READ);
exs_qhandle_t qh = exs_qcreate(10);

if (exs_send(fd, buf, bufsize, 0, mh, 0, qh) < 0) {
        perror("Could␣not␣start␣send␣operation");
        /* bail out */
}

/* do work in parallel with data transfer */

exs_event_t ev;
if (exs_qdequeue(qh, &ev, 1, NULL) < 0) {
        perror("Could␣not␣get␣send␣completion␣event");
        /* bail out */
}
fprintf(stderr, "Send␣of␣%d/%d␣bytes␣complete␣with␣errno=%d\n",
        bufsize, ev.exs_evt_union.exs_evt_xfer.exs_evt_length,
        ev.exs_evt_errno);
```

UNH EXS Protocol

Direct Transfer (SOCK_STREAM and SOCK_SEQPACKET)



Indirect Transfer (SOCK_STREAM only)



Outline

1. Background
2. RSockets
3. UNH EXS
4. Performance Evaluation
5. Conclusions
6. References

Performance Evaluation

- Comparison of TCP, rsockets, and EXS using **rstream** and **riostream** tools
  - rsockets rrecv()/rsend(): rstream
  - rsockets riomap()/riowrite(): riostream
  - TCP: rstream
  - EXS: rstream-like utility modified to take advantage of asynchronous operations
- No optimization done for rsocket and TCP cases
- Systems used: Intel Xeon 2.40 GHz E5-2609 CPUs, 64 GB RAM, PCI-e Gen 3
- HCAs: Mellanox ConnectX-3 FDR InfiniBand HCAs connected via Mellanox SX6036 FDR InfiniBand switch

Throughput Comparison

Implementing TCP Sockets over RDMA

MacArthur and Russell

Background

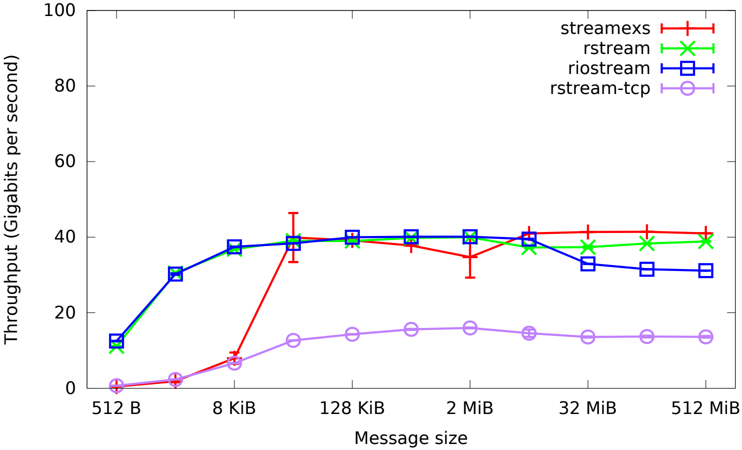
RSockets

UNH EXS

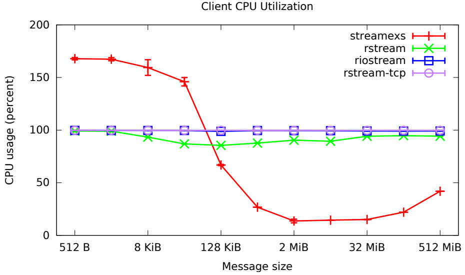
Performance Evaluation

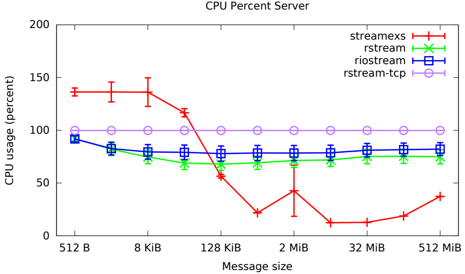
Conclusions

References

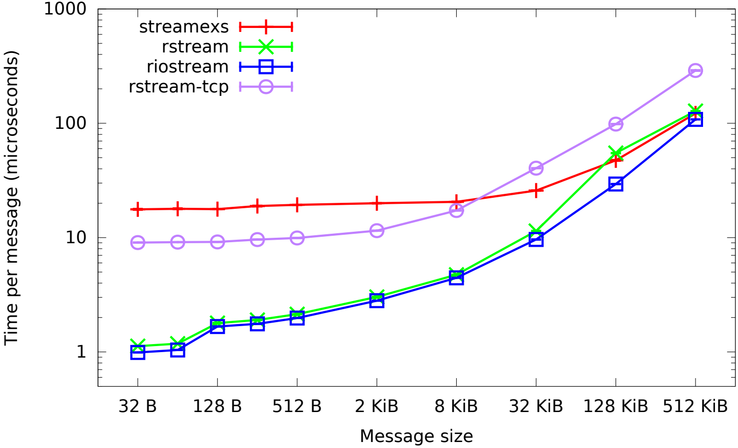


CPU Usage Comparison

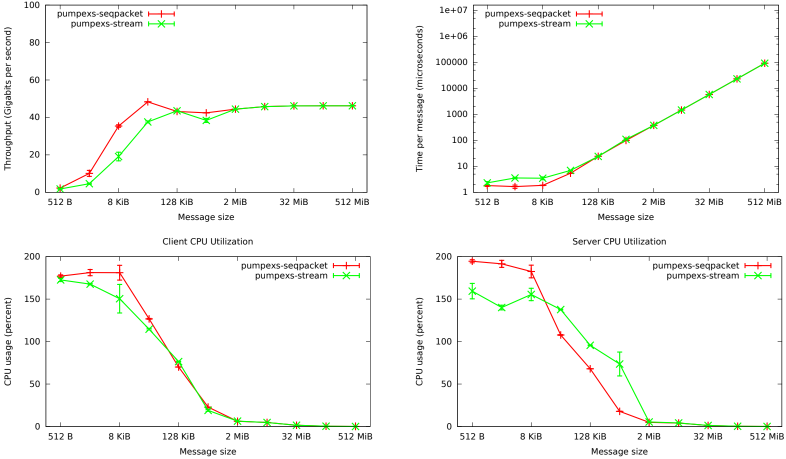




Latency Comparison



UNH EXS SOCK_SEQPACKET vs. SOCK_STREAM



Outline

1 Background

2 RSockets

3 UNH EXS

4 Performance Evaluation

5 Conclusions

6 References

Conclusions

- Socket buffering is implicit in sockets API
- Efficient zero-copy requires extensions to API
  - ES-API provides asynchronous operation
  - rsockets provides riomap()/riowrite()
- Message-oriented is more efficient to implement over RDMA than byte stream-oriented

Outline

1. Background
2. RSockets
3. UNH EXS
4. Performance Evaluation
5. Conclusions
6. References

References I

Infiniband Trade Association,
“Supplement to Infiniband Architecture Specification Volume 1, Release 1.2.1: Annex A4: Sockets Direct Protocol (SDP),”
Oct. 2011.

D. Goldenberg, M. Kagan, R. Ravid, and M. S. Tsirkin,
“Zero copy sockets direct protocol over Infiniband—preliminary implementation and performance analysis,”
in *High Performance Interconnects, 2005. Proceedings. 13th Symposium on.* IEEE, 2005, pp. 128–137.

References II

P. Balaji, S. Bhagvat, H.-W. Jin, and D. K. Panda, "Asynchronous zero-copy communication for synchronous sockets in the sockets direct protocol (SDP) over InfiniBand," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*. IEEE, 2006, pp. 8–pp.

Y. Lin, J. Han, J. Gao, and X. He, "uStream: a user-level stream protocol over InfiniBand," in *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*. IEEE, 2009, pp. 65–71.

References III

- S. Hefty,
  Rsockets.
  Available: https://www.openfabrics.org/ofa-documents/doc_download/495-rsockets.html

- Interconnect Software Consortium in association with the Open Group,
  "Extended Sockets API (ES-API) Issue 1.0,"
  Jan. 2005.

- ——,
  "A General-purpose API for iWARP and InfiniBand,"
  in *the First Workshop on Data Center Converged and Virtual Ethernet Switching (DC-CAVES)*, Sep. 2009.

References IV

Implementing TCP Sockets over RDMA

MacArthur and Russell

Background

RSockets

UNH EXS

Performance Evaluation

Conclusions

References

P. MacArthur and R. Russell,
"An Efficient Method for Stream Semantics over RDMA,"
in *Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2014)*,
May. 2014, to appear.