OPENFABRICS ALLIANCE

RSOCKETS

Sean Hefty

Intel Corporation

Motivation (AKA the Problem)

More Specifically…
Programming to Verbs

OPENFABRICS ALLIANCE

```
struct ibv_device **dev_list;
struct ibv_context *ib_ctx = NULL;
struct ibv_device_attr dev_attr;
struct ibv_port_attr port_attr;
int i, p, ret;

dev_list = ibv_get_device_list(NULL);
if (!dev_list)
    error();

for (i = 0; dev_list[i]; i++) {
    ib_ctx = ibv_open_device(dev_list[i]);
    if (!ib_ctx)
        error();

    ret = ibv_query_device(ib_ctx, &dev_attr)
    if (ret)
        error();
```

Get a list of devices and their attributes

More Specifically…

```
        for (p = 1; p < dev_attr.phys_port_cnt; p++) {
            ret = ibv_query_port(ib_ctx, i, &port_attr);
            if (ret)
                error();

            if (port_attr.state == IBV_PORT_ACTIVE)
                goto done;
        }
        ibv_close_device(dev_list[i]);
        ib_ctx = NULL;
}

done:
ibv_free_device_list(dev_list);
if (!ib_ctx)
        error();
```

Select a port and get its attributes

More Specifically…

```
struct ibv_pd *pd;
struct ibv_comp_channel *comp_channel;
struct ibv_cq *cq;

pd = ibv_alloc_pd(ib_ctx);
if (!pd)
    error();

comp_channel = ibv_create_comp_channel(ib_ctx);
if (!comp_channel)
    error();

cq = ibv_create_cq(ib_ctx, min(min(MY_SQ_SiZE + MY_RQ_SIZE),
                  dev_attr.max_qp_wr), dev_attr.max_cqe),
           NULL, comp_channel, 0);
if (!cq)
    error();
```

We need :

- protection domain
- completion channel
- completion queue

More Specifically…

```
struct ibv_qp *qp;
struct ibv_qp_init_attr qp_init_attr;

qp_init_attr.send_cq = cq;
qp_init_attr.recv_cq = cq;
qp_init_attr.cap.max_send_wr = min(MY_SQ_SIZE, dev_attr.max_qp_wr / 2);
qp_init_attr.cap.max_recv_wr = min(MY_RC_SIZE, dev_attr.max_qp_wr / 2);
qp_init_attr.cap.max_send_sge = min(MY_SQ_SGE, dev_attr.max_sge);
qp_init_attr.cap.max_recv_sge = min(MY_RQ_SGE, dev_attr.max_sge);
qp_init_attr.sq_sig_all = 1;
qp_init_attr.qp_context = NULL;
qp_init_attr.qp_type = IBV_QPT_RC;

qp = ibv_create_qp(pd, &qp_init_attr);
if (!qp)
    error();
```

- and a queue pair

More Specifically…

OPENFABRICS ALLIANCE

```
void *msgs;
struct ibv_mr *mr;
```

Allocate some messages to receive data…

```
msgs = calloc(qp_init_attr.cap.max_recv_wr, MY_MSG_SIZE);
if (!msgs)
    error();

mr = ibv_reg_mr(pd, msgs, qp_init_attr.cap.max_recv_wr * MY_MSG_SIZE,
        IBV_ACCESS_LOCAL_WRITE);
if (!mr)
    error();
```

and register them with the device

More Specifically…

```
struct ibv_recv_wr recv_wr, *bad_wr;
struct ibv_sge sge;

recv_wr.next = NULL;
recv_wr.sg_list = &sge;
recv_wr.num_sge = 1;
recv_wr.wr_id = 0;

sge.length = MY_MSG_SIZE;
sge.lkey = mr->lkey;
sge.addr = msgs;

for (i = 0; i < qp_init_attr.cap.max_recv_wr; i++) {
     ret = ibv_post_recv(qp, &recv_wr, &bad_wr);
     if (ret)
          error();

     sge.addr += MY_MSG_SIZE;
}
```

Post the messages on the queue pair *before* we connect

More Specifically…

I only have 30 minutes

assume we connect

and want to transfer data

More Specifically…

```
void *msg;
struct ibv_mr *mr;

msg = calloc(1, MY_MSG_SIZE);
if (!msg)
     error();

mr = ibv_reg_mr(pd, msg, MY_MSG_SIZE,
          IBV_ACCESS_LOCAL_WRITE);
if (!mr)
     error();
```

Allocate a send buffer…

and register it with the device

More Specifically…

```
struct ibv_send_wr send_wr, *bad_wr;
struct ibv_sge sge;

send_wr.next = NULL;
send_wr.sg_list = &sge;
send_wr.num_sge = 1;
send_wr.wr_id = 0;

sge.length = MY_MSG_SIZE;
sge.lkey = mr->lkey;
sge.addr = msgs;

<format_msg(msgs, 0);>

ret = ibv_post_send(qp, &send_wr, &bad_wr);
if (ret)
    error();
```

All this just to send?

More Specifically…

OPENFABRICS ALLIANCE

```
struct ibv_wc wc;
struct ibv_cq *cq;
void *context;
int ret;

do {
     ret = ibv_poll_cq(cq, 1, &wc);
     if (ret)
          break;

     ret = ibv_req_notify_cq(cq, 0);
     if (ret)
          error();

     ret = ibv_poll_cq(cq, 1, &wc);
     if (ret)
          break;
```

Wait for the send to complete or we receive a response

Remember to poll the completion queue after requesting notification

More Specifically…

```
        ret = ibv_get_cq_event(comp_channel, &cq, &context);
        if (ret)
            error();

        ibv_ack_cq_events(cq, 1);
    } while (1);

    if (ret < 0)
        error();
```

Wait for an event and check the completion queue again

And it's just that easy to send data!

Motivation continued

- And it's just as bad on the receive side

Now, anyone want to DO an actual RDMA operation?

Motivation continued

Actually, I just wanted to echo typing between two systems connected by IB that did not have ipoib (or sdp) but this wouldn't make as good an intro

Big Intro… RSOCKETS!

OPENFABRICS ALLIANCE

Ta-da!

- RDMA sockets API
  - Another API - ~joy~
- Calls that look and behave *like sockets*
- Connects *like sockets*
- Byte streaming transfers *like sockets*
  - I.e. SOCK_STREAM
- Support for nonblocking operation *like sockets*

Like sockets … except that it's not

Goals

Support well-known network programming concepts

- Socket programming *concepts* with minimal to no need to learn *anything* about RDMA
  - Let's face it, no matter how many APIs we create developers will still learn sockets
  - Sockets will continue as the common fallback API
- Support existing socket applications under ideal conditions
- ***SDP license free!***

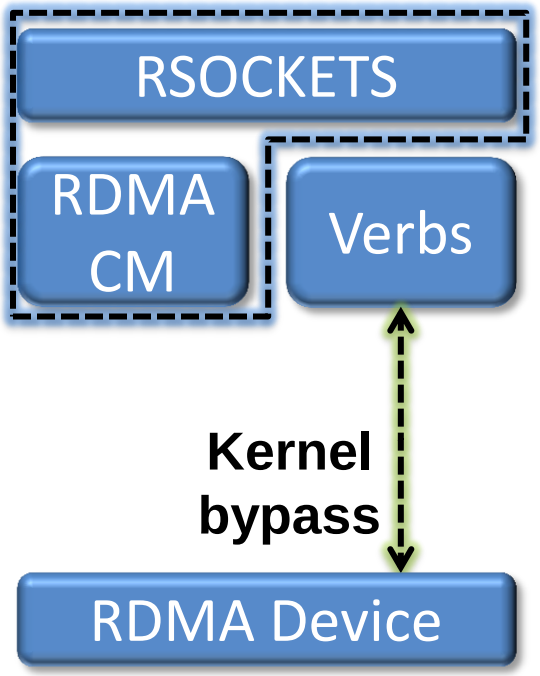
Goals

OPENFABRICS ALLIANCE

High performance

- Outperform ipoib (and sdp)
  - Or it's pointless, except for limited environments
- Perform favorably compared to native RDMA implementation
  - Or there's not a strong enough reason NOT to learn RDMA programming
  - Narrow the cost-benefit gap of maintaining verbs support in an application long term

RSOCKETS Overview



- Proprietary protocol / algorithm
  - I made it up
  - Will be open sourced
- Entirely user-space implementation
  - Well, if we ignore the existing RDMA support
  - No need to merge anything upstream!

R + SOCKET Interface

OPENFABRICS ALLIANCE

Connections	• rsocket, rbind, rlisten, raccept, rconnect • rshutdown, rclose
Data transfers	• rrecv, rrecvfrom, rrecvmsg, rread, rreadv • rsend, rsendto, rsendmsg, rwrite, rwritev
Asynchronous support	• rpoll, rselect
Socket options	• rsetsockopt, rgetsockopt, rfcntl
Other useful calls	• rgetpeername, rgetsockname

Supported Features

Implementation based on needs of OSU and Intel MPI

Functions take same parameters as sockets

- PF_INET, PF_INET6, SOCK_STREAM, IPPROTO_TCP
- MSG_DONTWAIT, MSG_PEEK
- SO_REUSEADDR, TCP_NODELAY, SO_ERROR
- SO_SNDBUF, SO_RCVBUF
- O_NONBLOCK

Now a word from our sponsor…

More words from our sponsor…

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

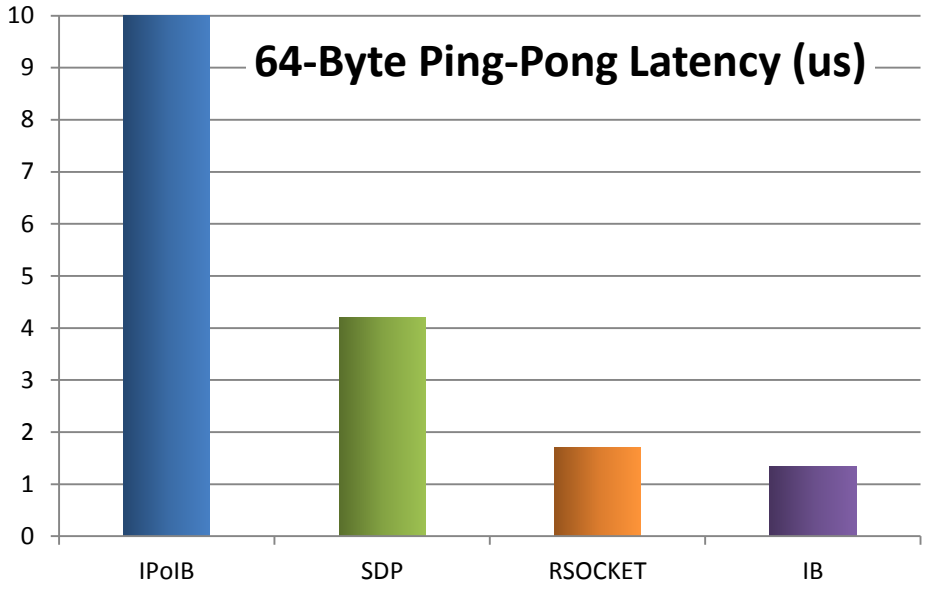
Notice revision #20110804

- 8-node Xeon X5570 @ 2.93 Ghz (Nehalem) cluster
- 8 cores / node
- 40 Gbps Infiniband
- 2 node latency and BW tests rstream / perftest
- 64 process MPI runs

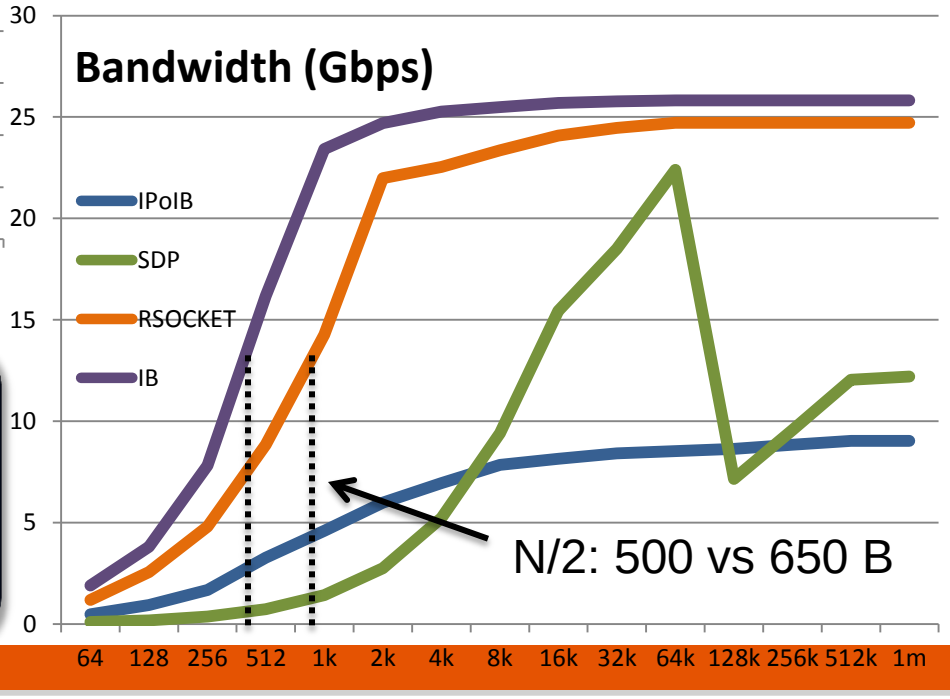
What's the Performance?

OPENFABRICS ALLIANCE

Note: implementation has minimal optimizations

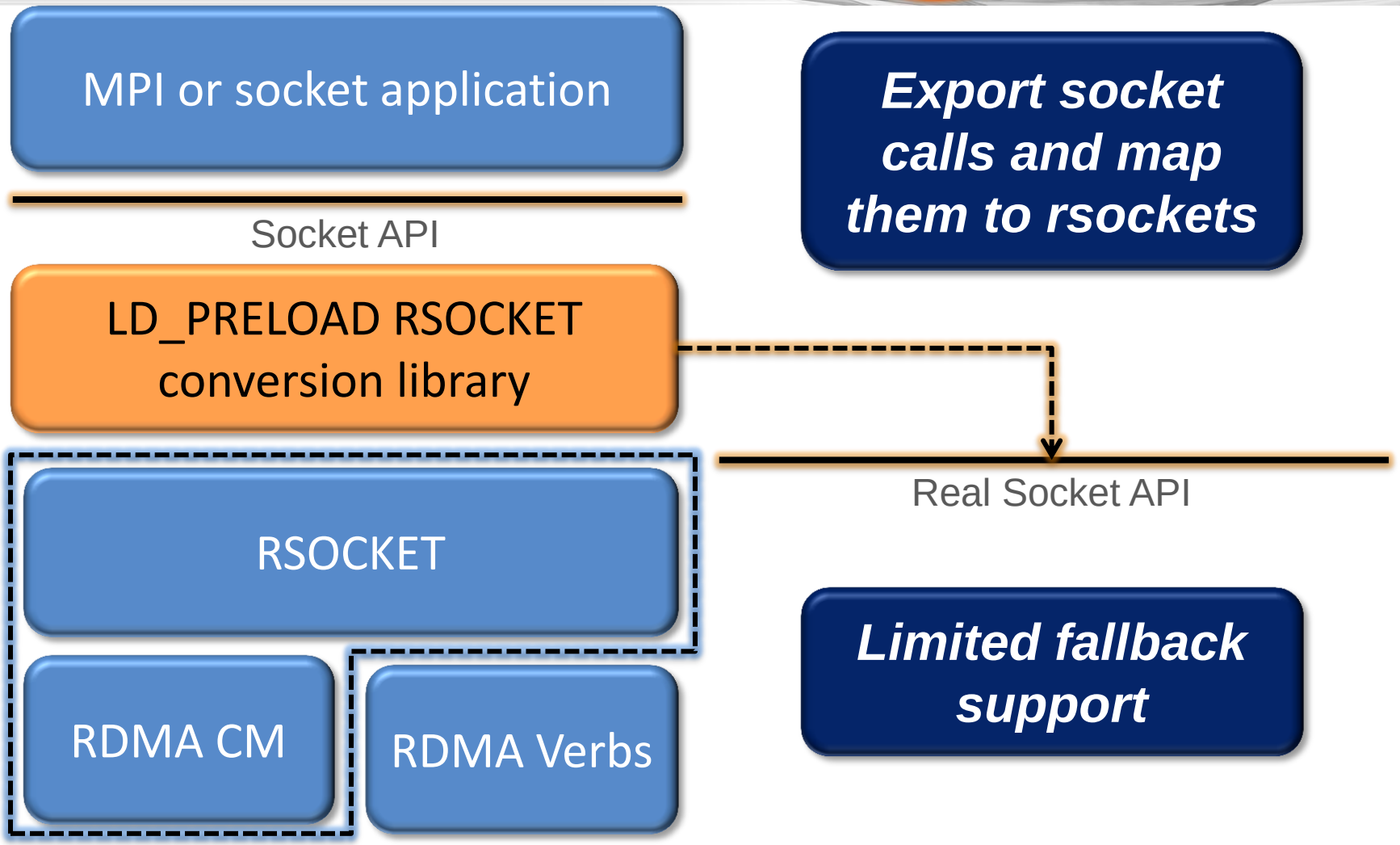


Promising latency and bandwidth



Can it work with existing apps? At all? Well?

Supporting Existing Apps

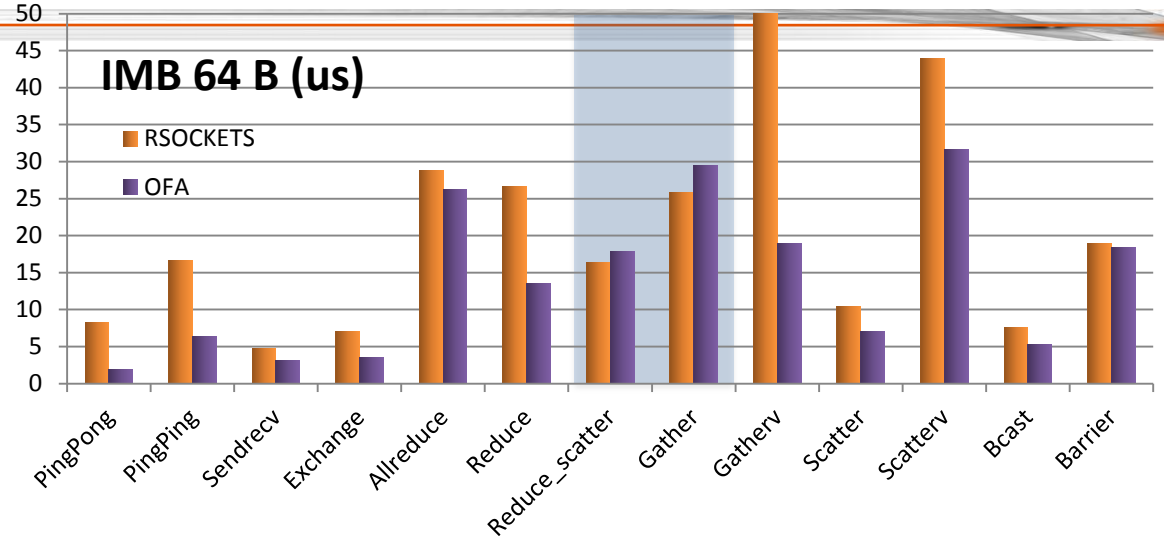


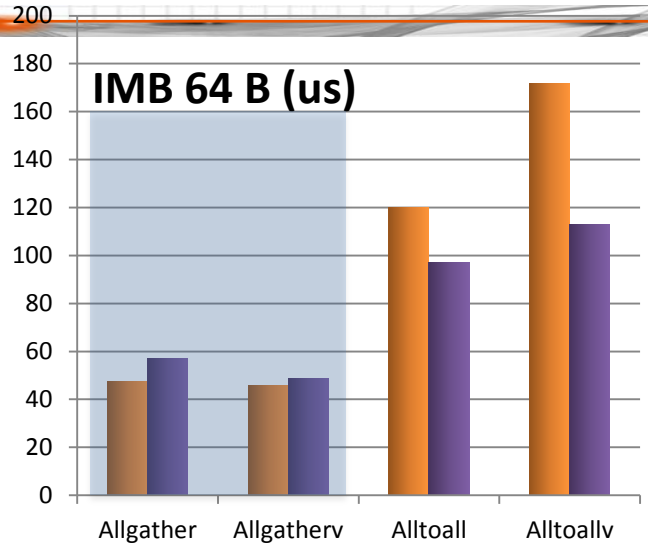
IMB - Intel MPI Benchmarks

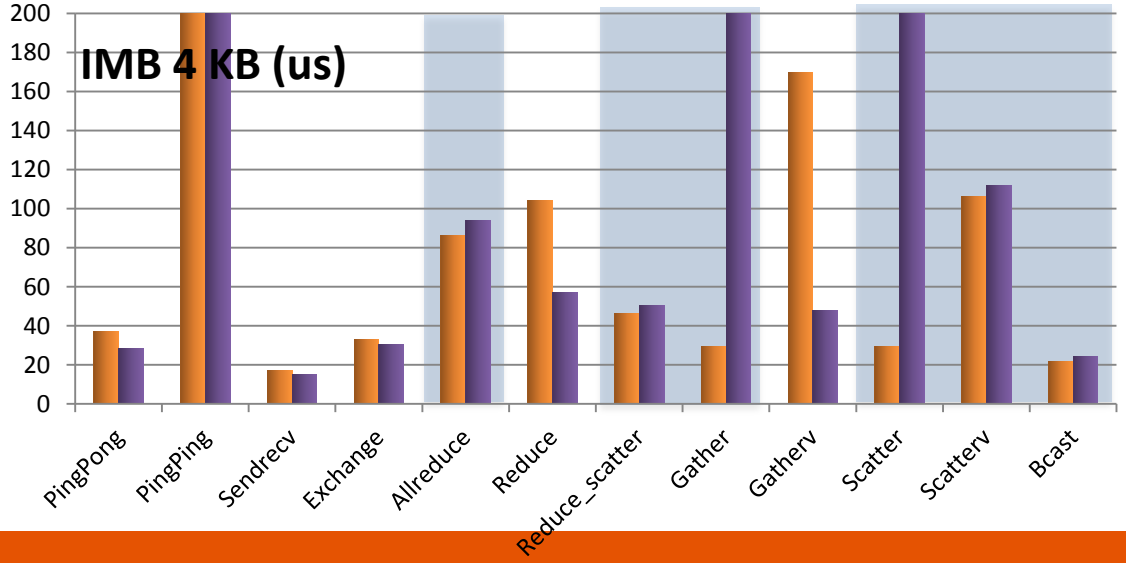
- Measure important MPI functionality
- Results for arbitrarily selected sizes
- IPoIB performance was much worse
  - Omitted for space
- SDP tests failed for 64 ranks
  - Had lower performance for fewer ranks

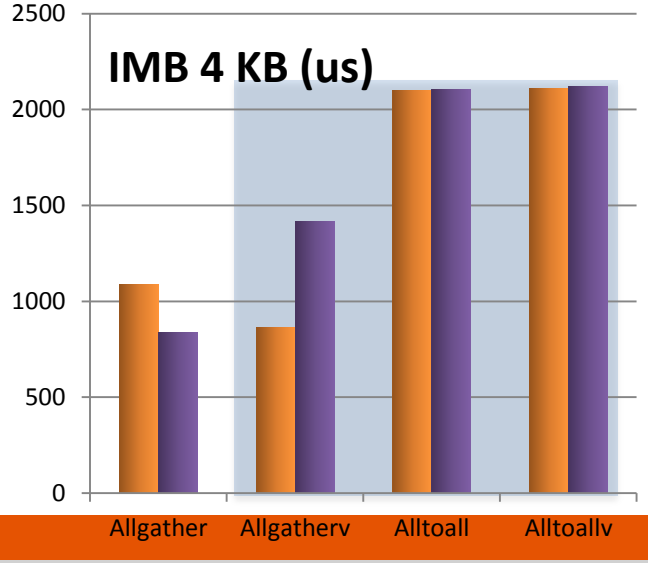
Results in microseconds - lower is better

IMB Results

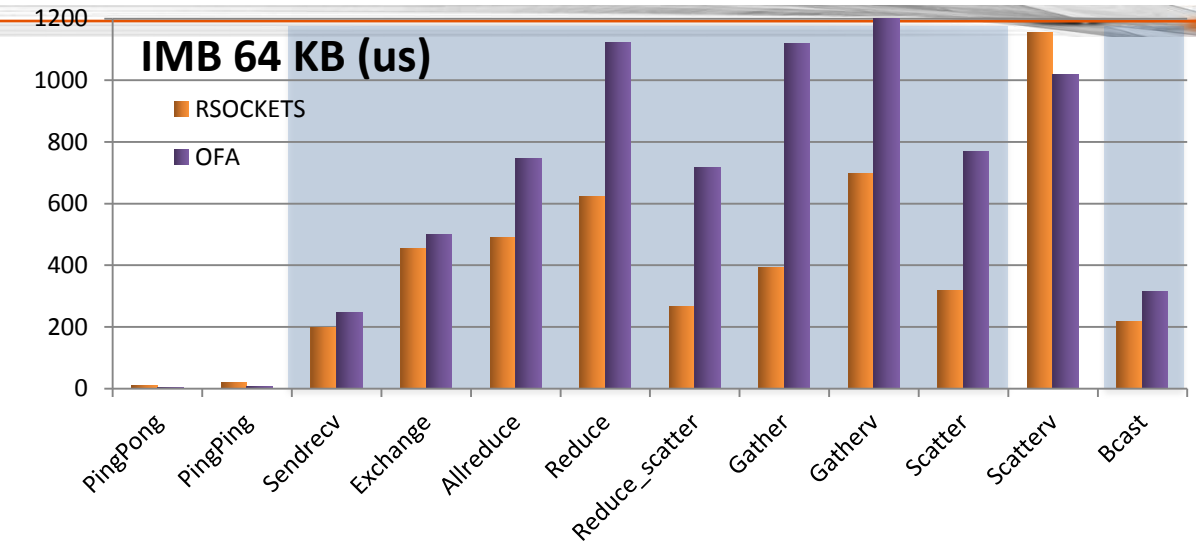

IMB 64 B (us)
RSOCKETS
OFA
50
45
40
35
30
25
20
15
10
5
0
PingPong
PingPing
Sendrecv
Exchange
Allreduce
Reduce
Reduce_scatter
Gather
Gatherv
Scatter
Scatterv
Bcast
Barrier

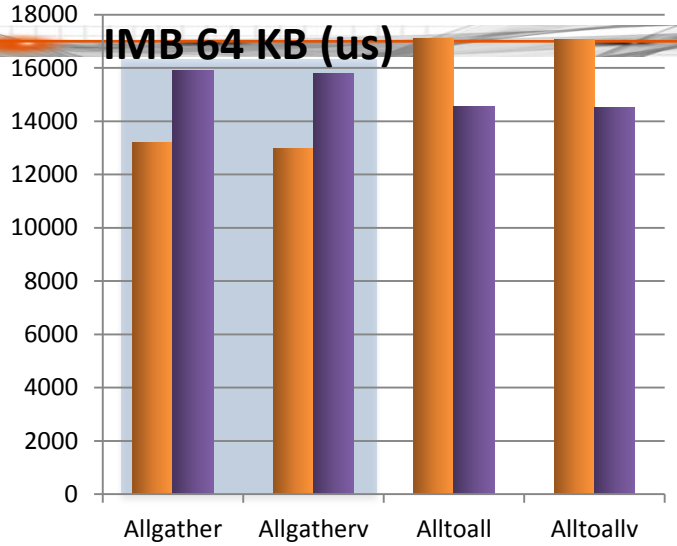

IMB 64 B (us)
200
180
160
140
120
100
80
60
40
20
0
Allgather
Allgatherv
Alltoall
Alltoallv

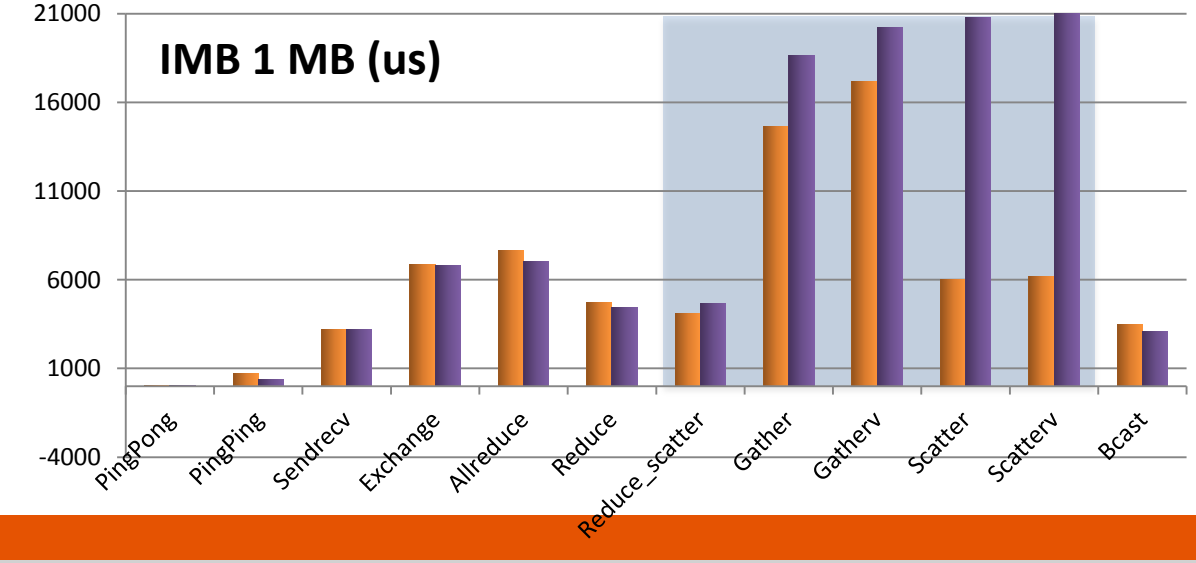

IMB 4 KB (us)
200
180
160
140
120
100
80
60
40
20
0
PingPong
PingPing
Sendrecv
Exchange
Allreduce
Reduce
Reduce_scatter
Gather
Gatherv
Scatter
Scatterv
Bcast

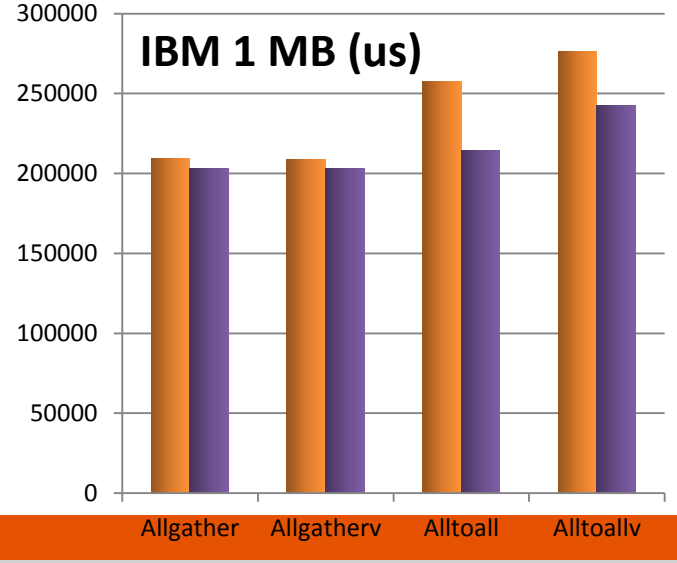

IMB 4 KB (us)
2500
2000
1500
1000
500
0
Allgather
Allgatherv
Alltoall
Alltoallv

IMB Results


IMB 64 KB (us)
RSOCKETS
OFA
1200
1000
800
600
400
200
0
PingPong
PingPing
Sendrecv
Exchange
Allreduce
Reduce
Reduce_scatter
Gather
Gatherv
Scatter
Scatterv
Bcast


IMB 64 KB (us)
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Allgather
Allgatherv
Alltoall
Alltoallv

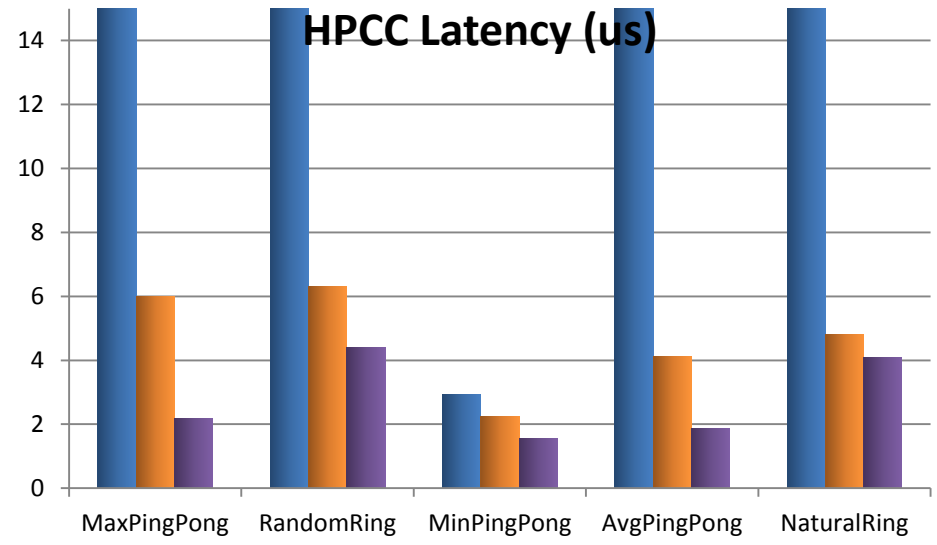

IMB 1 MB (us)
21000
16000
11000
6000
1000
-4000
PingPong
PingPing
Sendrecv
Exchange
Allreduce
Reduce
Reduce_scatter
Gather
Gatherv
Scatter
Scatterv
Bcast

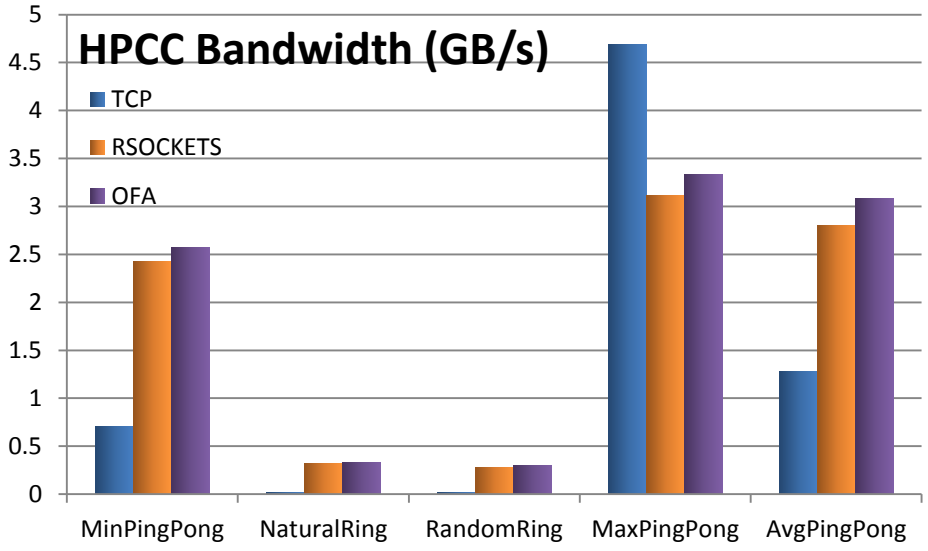

IBM 1 MB (us)
300000
250000
200000
150000
100000
50000
0
Allgather
Allgatherv
Alltoall
Alltoallv

What About a “Real” App?

- HPC Challenge benchmarks
  - Set of higher-level benchmarks
- As close to a “real” app that I could easily run
- Selected results reported
  - SDP failed to run
  - IPoIB results included

HPC Challenge



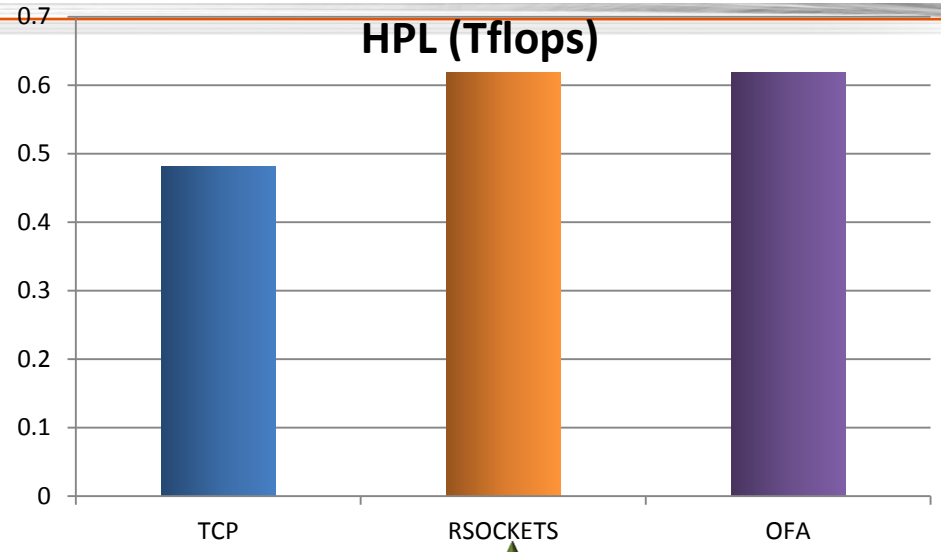


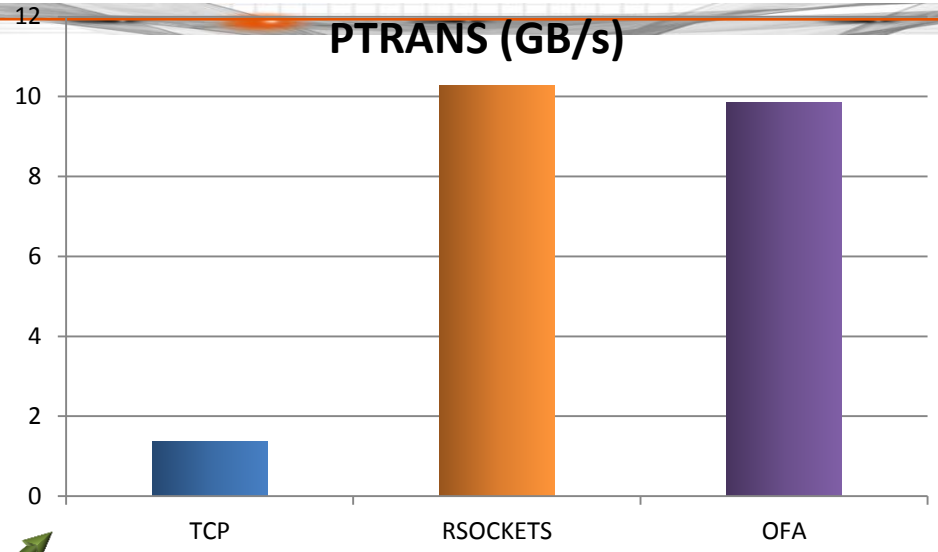
Lower is better

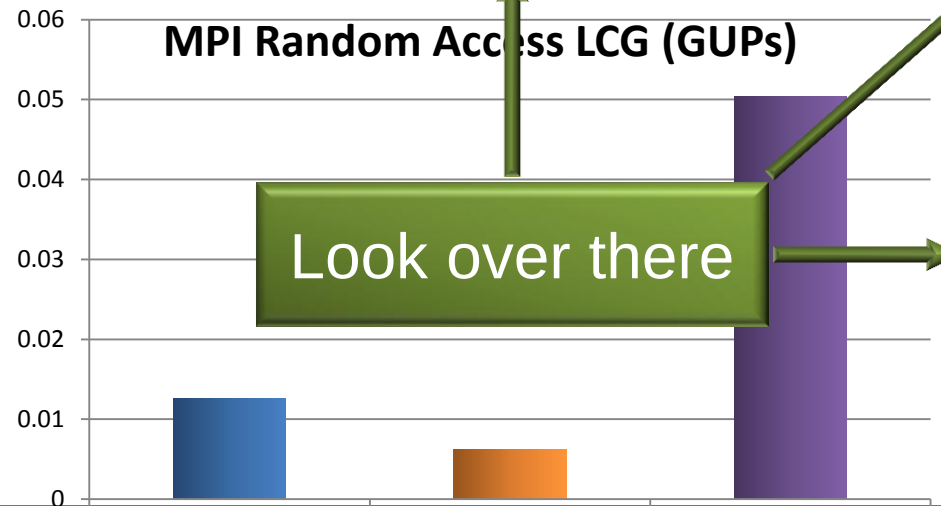
Higher is better

HPC Challenge

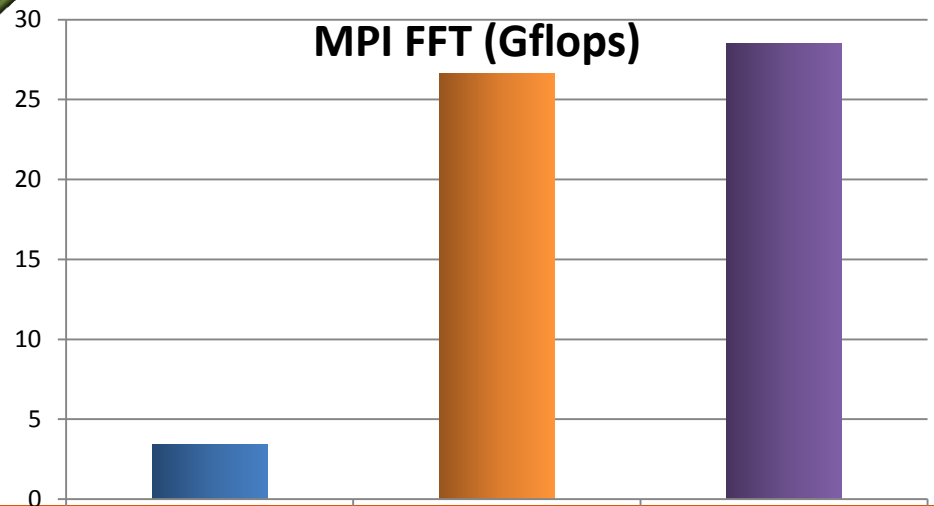
Higher is better







Look over there



Closing the Performance Gap

- Notable area for improvement:
  - Direct data placement (reduce memory copies)
- Possible, but…
- Most target applications use *nonblocking* sockets
  - Restricts use with recv()
  - Which reduces usefulness with send()
- Alternatives?

Closing the Performance Gap

- Is there any way to add direct access to RDMA operations through sockets?
  - Get that last bit of performance
- While keeping it simple?
- And.. without actually needing to know anything about RDMA?
  - Or these acronyms: PD, CQ, HCA, MR, QP, LID, GID, …
- And make it generic, so that other technologies may be able to use it
  - Tag matching, file I/O, SSDs
- And continue to support the socket programming model!

Direct Data Placement Extensions

This is a discussion point only

- Can we find calls that *blend* in with existing calls?
- Now we *may be* talking about new programming concepts
- Are there any existing calls that are usable?
  - send, sendto, sendmsg, write, writev, pwrite …
  - recv, recvfrom, recvmsg, read, readv, pread …
  - mmap, lseek, fseek, fgetpos, fsetpos, fsync …

Although not used with sockets, these calls may be used as guides

Direct Data Placement APIs

rmmap	• Map memory *to* a specified offset • Specify access restrictions • Maps to memory registration
rget	• Read from an offset into a local buffer • Maps to RDMA read operation
rput	• Write from a local buffer to the given offset • Maps to RDMA write operation

Direct Data Placement

- *Extends* current usage model
  - No change to connecting or send/recv calls
  - Memory region data exchanged underneath
- Appears usable for multiple technologies
- Seems easy to learn and use

The Real Problem

Target applications use *nonblocking* sockets

Direct data placement calls may not block

Notification of completion should come from select() and poll() calls

Would need to determine how to handle nonblocking calls without an indecent exposure to RDMA

Requests to Verbs

- Asynchronous memory registration
  - Assist with direct data placement
- A single file descriptor for all RDMA resources
  - Event queue, completion queue, connections
  - Simplifies implementation
- Way to transfer control of a set of RDMA resources to another process
  - Help support apps that fork

What's Your Opinion?

OPENFABRICS ALLIANCE

Does rsockets have a place going forward?	What's the best way to add direct data placement?	What other features are worth implementing?
• It's really 5 years too late • In limited environments • Absolutely	• Not at all • Best solution using existing socket calls • Extensions	• Datagram support? • Out of band data? • Fork?

```
thank_you;
exit(0);
```