Writing RDMA applications on Linux

Example programs

Roland Dreier
rolandd@cisco.com

September 3, 2007

1 Client (active) example

```
/*
 * build:
 *    cc -o client client.c -lrdmacm
 *
 * usage:
 *    client <servername> <val1> <val2>
 *
 * connects to server, sends val1 via RDMA write and val2 via send,
 * and receives val1+val2 back from the server.
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>

#include <infiniband/arch.h>
#include <rdma/rdma_cma.h>

enum {
        RESOLVE_TIMEOUT_MS = 5000,
};

struct pdata {
        uint64_t        buf_va;
```

```
        uint32_t        buf_rkey;
};

int main(int argc, char *argv[])
{
        struct pdata                    server_pdata;

        struct rdma_event_channel   *cm_channel;
        struct rdma_cm_id           *cm_id;
        struct rdma_cm_event        *event;
        struct rdma_conn_param       conn_param = { };

        struct ibv_pd               *pd;
        struct ibv_comp_channel     *comp_chan;
        struct ibv_cq               *cq;
        struct ibv_cq               *evt_cq;
        struct ibv_mr               *mr;
        struct ibv_qp_init_attr      qp_attr = { };
        struct ibv_sge               sge;
        struct ibv_send_wr           send_wr = { };
        struct ibv_send_wr          *bad_send_wr;
        struct ibv_recv_wr           recv_wr = { };
        struct ibv_recv_wr          *bad_recv_wr;
        struct ibv_wc                wc;
        void                        *cq_context;

        struct addrinfo             *res, *t;
        struct addrinfo              hints = {
                .ai_family    = AF_INET,
                .ai_socktype = SOCK_STREAM
        };
        int                          n;

        uint32_t                    *buf;

        int                          err;

        /* Set up RDMA CM structures */

        cm_channel = rdma_create_event_channel();
        if (!cm_channel)
                return 1;

        err = rdma_create_id(cm_channel, &cm_id, NULL, RDMA_PS_TCP);
        if (err)
                return err;

        n = getaddrinfo(argv[1], "20079", &hints, &res);
        if (n < 0)
```

```
        return 1;
                                                                    80
/* Resolve server address and route */

for (t = res; t; t = t->ai_next) {
        err = rdma_resolve_addr(cm_id, NULL, t->ai_addr,
                                RESOLVE_TIMEOUT_MS);
        if (!err)
                break;
}
if (err)
        return err;                                                 90

err = rdma_get_cm_event(cm_channel, &event);
if (err)
        return err;

if (event->event != RDMA_CM_EVENT_ADDR_RESOLVED)
        return 1;

rdma_ack_cm_event(event);
                                                                    100
err = rdma_resolve_route(cm_id, RESOLVE_TIMEOUT_MS);
if (err)
        return err;

err = rdma_get_cm_event(cm_channel, &event);
if (err)
        return err;

if (event->event != RDMA_CM_EVENT_ROUTE_RESOLVED)
        return 1;                                                   110

rdma_ack_cm_event(event);

/* Create verbs objects now that we know which device to use */

pd = ibv_alloc_pd(cm_id->verbs);
if (!pd)
        return 1;

comp_chan = ibv_create_comp_channel(cm_id->verbs);                  120
if (!comp_chan)
        return 1;

cq = ibv_create_cq(cm_id->verbs, 2, NULL, comp_chan, 0);
if (!cq)
        return 1;
```

```
if (ibv_req_notify_cq(cq, 0))
        return 1;
                                                                        130
buf = calloc(2, sizeof (uint32_t));
if (!buf)
        return 1;

mr = ibv_reg_mr(pd, buf, 2 * sizeof (uint32_t), IBV_ACCESS_LOCAL_WRITE);
if (!mr)
        return 1;

qp_attr.cap.max_send_wr = 2;
qp_attr.cap.max_send_sge = 1;                                           140
qp_attr.cap.max_recv_wr = 1;
qp_attr.cap.max_recv_sge = 1;

qp_attr.send_cq          = cq;
qp_attr.recv_cq          = cq;

qp_attr.qp_type          = IBV_QPT_RC;

err = rdma_create_qp(cm_id, pd, &qp_attr);
if (err)                                                                150
        return err;

conn_param.initiator_depth = 1;
conn_param.retry_count   = 7;

/* Connect to server */

err = rdma_connect(cm_id, &conn_param);
if (err)
        return err;                                                     160

err = rdma_get_cm_event(cm_channel, &event);
if (err)
        return err;

if (event->event != RDMA_CM_EVENT_ESTABLISHED)
        return 1;

memcpy(&server_pdata, event->param.conn.private_data,
       sizeof server_pdata);                                            170

rdma_ack_cm_event(event);

/* Prepost receive */

sge.addr   = (uintptr_t) buf;
```

```
sge.length = sizeof (uint32_t);
sge.lkey   = mr->lkey;

recv_wr.wr_id   = 0;                                                    180
recv_wr.sg_list = &sge;
recv_wr.num_sge = 1;

if (ibv_post_recv(cm_id->qp, &recv_wr, &bad_recv_wr))
        return 1;

/* Write/send two integers to be added */

buf[0] = strtoul(argv[2], NULL, 0);
buf[1] = strtoul(argv[3], NULL, 0);                                     190

printf("%d + %d = ", buf[0], buf[1]);

buf[0] = htonl(buf[0]);
buf[1] = htonl(buf[1]);

sge.addr   = (uintptr_t) buf;
sge.length = sizeof (uint32_t);
sge.lkey   = mr->lkey;
                                                                        200
send_wr.wr_id               = 1;
send_wr.opcode              = IBV_WR_RDMA_WRITE;
send_wr.sg_list             = &sge;
send_wr.num_sge             = 1;
send_wr.wr.rdma.rkey        = ntohl(server_pdata.buf_rkey);
send_wr.wr.rdma.remote_addr = ntohll(server_pdata.buf_va);

if (ibv_post_send(cm_id->qp, &send_wr, &bad_send_wr))
        return 1;
                                                                        210
sge.addr   = (uintptr_t) buf + sizeof (uint32_t);
sge.length = sizeof (uint32_t);
sge.lkey   = mr->lkey;

send_wr.wr_id               = 2;
send_wr.opcode              = IBV_WR_SEND;
send_wr.send_flags          = IBV_SEND_SIGNALED;
send_wr.sg_list             = &sge;
send_wr.num_sge             = 1;
                                                                        220
if (ibv_post_send(cm_id->qp, &send_wr, &bad_send_wr))
        return 1;

/* Wait for receive completion */
```

```
        while (1) {
                if (ibv_get_cq_event(comp_chan, &evt_cq, &cq_context))
                        return 1;

                if (ibv_req_notify_cq(cq, 0))                                   230
                        return 1;

                if (ibv_poll_cq(cq, 1, &wc) != 1)
                        return 1;

                if (wc.status != IBV_WC_SUCCESS)
                        return 1;

                if (wc.wr_id == 0) {
                        printf("%d\n", ntohl(buf[0]));                          240
                        return 0;
                }
        }

        return 0;
}
```

2 Server (passive) example

```
/*
 * build:
 *    cc -o server server.c -lrdmacm
 *
 * usage:
 *    server
 *
 * waits for client to connect, receives two integers, and sends their
 * sum back to the client.
 */                                                                             10

#include <stdlib.h>
#include <stdint.h>
#include <arpa/inet.h>

#include <infiniband/arch.h>
#include <rdma/rdma_cma.h>

enum {
        RESOLVE_TIMEOUT_MS = 5000,                                              20
};
```

```
struct pdata {
        uint64_t        buf_va;
        uint32_t        buf_rkey;
};

int main(int argc, char *argv[])
{
        struct pdata                    rep_pdata;

        struct rdma_event_channel       *cm_channel;
        struct rdma_cm_id               *listen_id;
        struct rdma_cm_id               *cm_id;
        struct rdma_cm_event            *event;
        struct rdma_conn_param          conn_param = { };

        struct ibv_pd                   *pd;
        struct ibv_comp_channel         *comp_chan;
        struct ibv_cq                   *cq;
        struct ibv_cq                   *evt_cq;
        struct ibv_mr                   *mr;
        struct ibv_qp_init_attr         qp_attr = { };
        struct ibv_sge                  sge;
        struct ibv_send_wr              send_wr = { };
        struct ibv_send_wr              *bad_send_wr;
        struct ibv_recv_wr              recv_wr = { };
        struct ibv_recv_wr              *bad_recv_wr;
        struct ibv_wc                   wc;
        void                            *cq_context;

        struct sockaddr_in              sin;

        uint32_t                        *buf;

        int                             err;

        /* Set up RDMA CM structures */

        cm_channel = rdma_create_event_channel();
        if (!cm_channel)
                return 1;

        err = rdma_create_id(cm_channel, &listen_id, NULL, RDMA_PS_TCP);
        if (err)
                return err;

        sin.sin_family      = AF_INET;
        sin.sin_port        = htons(20079);
        sin.sin_addr.s_addr = INADDR_ANY;
```

```
/* Bind to local port and listen for connection request */

err = rdma_bind_addr(listen_id, (struct sockaddr *) &sin);
if (err)
        return 1;

err = rdma_listen(listen_id, 1);
if (err)
        return 1;                                                    80

err = rdma_get_cm_event(cm_channel, &event);
if (err)
        return err;

if (event->event != RDMA_CM_EVENT_CONNECT_REQUEST)
        return 1;

cm_id = event->id;
                                                                     90
rdma_ack_cm_event(event);

/* Create verbs objects now that we know which device to use */

pd = ibv_alloc_pd(cm_id->verbs);
if (!pd)
        return 1;

comp_chan = ibv_create_comp_channel(cm_id->verbs);
if (!comp_chan)                                                     100
        return 1;

cq = ibv_create_cq(cm_id->verbs, 2, NULL, comp_chan, 0);
if (!cq)
        return 1;

if (ibv_req_notify_cq(cq, 0))
        return 1;

buf = calloc(2, sizeof (uint32_t));                                 110
if (!buf)
        return 1;

mr = ibv_reg_mr(pd, buf, 2 * sizeof (uint32_t),
                IBV_ACCESS_LOCAL_WRITE |
                IBV_ACCESS_REMOTE_READ |
                IBV_ACCESS_REMOTE_WRITE);
if (!mr)
        return 1;
                                                                    120
```

```
qp_attr.cap.max_send_wr = 1;
qp_attr.cap.max_send_sge = 1;
qp_attr.cap.max_recv_wr = 1;
qp_attr.cap.max_recv_sge = 1;

qp_attr.send_cq          = cq;
qp_attr.recv_cq          = cq;

qp_attr.qp_type          = IBV_QPT_RC;
                                                                    130
err = rdma_create_qp(cm_id, pd, &qp_attr);
if (err)
        return err;

/* Post receive before accepting connection */

sge.addr   = (uintptr_t) buf + sizeof (uint32_t);
sge.length = sizeof (uint32_t);
sge.lkey   = mr->lkey;
                                                                    140
recv_wr.sg_list = &sge;
recv_wr.num_sge = 1;

if (ibv_post_recv(cm_id->qp, &recv_wr, &bad_recv_wr))
        return 1;

rep_pdata.buf_va  = htonll((uintptr_t) buf);
rep_pdata.buf_rkey = htonl(mr->rkey);

conn_param.responder_resources = 1;                                 150
conn_param.private_data      = &rep_pdata;
conn_param.private_data_len  = sizeof rep_pdata;

/* Accept connection */

err = rdma_accept(cm_id, &conn_param);
if (err)
        return 1;

err = rdma_get_cm_event(cm_channel, &event);                        160
if (err)
        return err;

if (event->event != RDMA_CM_EVENT_ESTABLISHED)
        return 1;

rdma_ack_cm_event(event);

/* Wait for receive completion */
```

```
                                                                    170
        if (ibv_get_cq_event(comp_chan, &evt_cq, &cq_context))
                return 1;

        if (ibv_req_notify_cq(cq, 0))
                return 1;

        if (ibv_poll_cq(cq, 1, &wc) < 1)
                return 1;

        if (wc.status != IBV_WC_SUCCESS)                            180
                return 1;

        /* Add two integers and send reply back */

        buf[0] = htonl(ntohl(buf[0]) + ntohl(buf[1]));

        sge.addr   = (uintptr_t) buf;
        sge.length = sizeof (uint32_t);
        sge.lkey   = mr->lkey;
                                                                    190
        send_wr.opcode   = IBV_WR_SEND;
        send_wr.send_flags = IBV_SEND_SIGNALED;
        send_wr.sg_list    = &sge;
        send_wr.num_sge = 1;

        if (ibv_post_send(cm_id->qp, &send_wr, &bad_send_wr))
                return 1;

        /* Wait for send completion */
                                                                    200
        if (ibv_get_cq_event(comp_chan, &evt_cq, &cq_context))
                return 1;

        if (ibv_poll_cq(cq, 1, &wc) < 1)
                return 1;

        if (wc.status != IBV_WC_SUCCESS)
                return 1;

        ibv_ack_cq_events(cq, 2);                                   210

        return 0;
}
```

3 More information

- `http://www.openfabrics.org/`
  packages and git trees for most userspace components
- `http://www.infinibandta.org/specs/`
  sometimes going to the spec is the best way to get a precise answer
- `mailto:general@lists.openfabrics.org`
  the best place for getting help from other developers