EVALUATING MULTIPLE ENDPOINTS FOR MPI WITH LIBIBVERBS

Rohit Zambre*, Abdelhalim Amer†, Aparna Chandramowlishwaran*, Pavan Balaji†

*University of California, Irvine
†Argonne National Lab

THE STATUS QUO

- *Network hardware today*
  - Feature multiple communication contexts.
  - Continuing to move in this direction by featuring multiple NICs on a node.
  - Expose parallelism at the network hardware level.
- *MPI today*
  - Each process uses only one communication context.
  - Multiple communication contexts used only if multiple ranks exist on a node.
  - Even in hybrid environments, all threads use the context available to one rank.
- *Underutilized resources*
  - When number of ranks on a node is smaller than the number of contexts.
  - MPI rank could acquire the remaining resources and tap the network parallelism.
- *Scenario to study*
  - A single process sending messages using multiple network resources.
  - Evaluation over InfiniBand.
  - Using libibverbs.

BENCHMARK DESIGN

Sender

```
int posts_per_qp = messages / num_ctxs;
MPI_Barrier(MPI_COMM_WORLD);
/* Sender loop */
while (tot_post_count<messages || tot_comp_count<messages) {
  for (i=0; i<num_ctxs; i++) {
    /* Post */
    posts = min(posts_per_qp - post_count[i],
                qp_depth - (post_count[i]-comp_count[i]));
    for (k=0; k<posts; k++) {
    ibv_post_send(qp[i], &send_wqe[i], &bad_send_wqe);
    }
    post_count[i] += posts;
    tot_post_count += posts;

    /*Poll*/
    if (comp_count[i]<posts_per_qp) {
      cqe_count = ibv_poll_cq(cq[i], num_comps, WC);
      comp_count[i] += cqe_count;
      tot_comp_count += cqe_count;
    }
  }
}
```

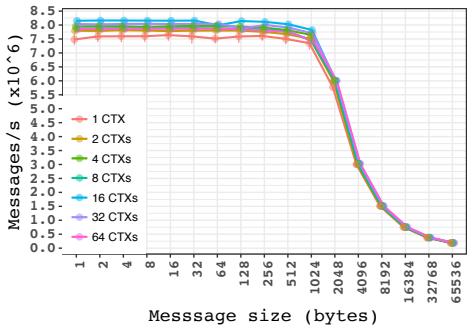
Receiver

```
int posts_per_qp = messages / num_ctxs;
MPI_Barrier(MPI_COMM_WORLD);
/* Receiver loop */
while (tot_comp_count<messages) {
  for (i=0; i<num_ctxs; i++) {
    cqe_count = ibv_poll_cq(cq[i], num_comps, WC);
    if (cqe_count>0) {
      for(j=0; j<cqe_count; j++) {
        if (post_count[i]<posts_per_qp) {
          ibv_post_recv(qp[i], &recv_wqe[i],
                        &bad_recv_wqe);
        }
        ++post_count[i];
      }
    } } }
```

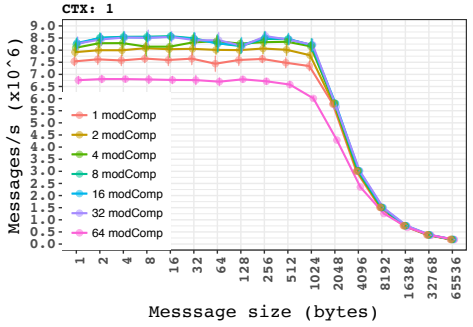
MESSAGE RATES

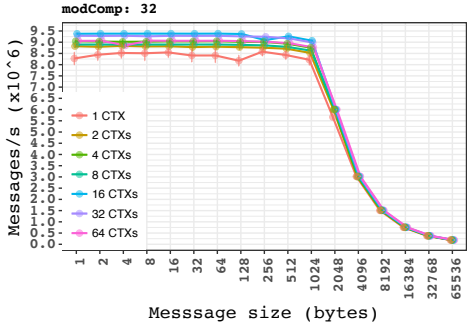
- A sender-receiver pair on the JLSE Cluster
- 64-core Intel Xeon E7-8867 CPU @ 2.5 GHz; RHEL 7.4
- Mellanox OFED 4.1-1.0.2.0

Completions every issue



Batch completions



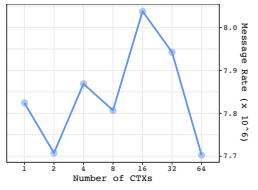


EVALUATION

- *Explaining observed Message Rates*
  - Contingent on rate of issuing `ibv_post_sends`.
  - `ibv_post_send` called every 296.18 cycles in the post-loop
  - But `ibv_post_send` of next post-loop is called only after 3838.88 cycles
  - Hence, bottlenecked by completion-check of issued sends
- *A **completed send***
  - A send that involves a `ibv_post_send` and a successful `ibv_poll_cq`
- *Issue-rate of completed sends*

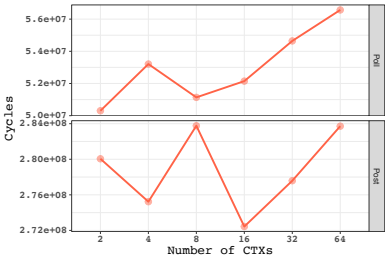
Completed-send cycles

Calculated MRs



- *Network processing saturates with increase in number of CTXs*

- *While the aggregate number of posting and polling cycles remain the same with the same number of CTXs, the slow down fro 16 CTXs to 64 CTXs is due to the increase in number of polls*



CONCLUSION & NEXT STEPS

- ***Network hardware parallelism (multiple contexts) can help improve performance to some extent.***
  - but should be used judiciously.
- ***Next steps***
  - Evaluate on different high-performance interconnects and study similarities and differences of behavior with libibverbs
  - Evaluate with virtual topologies and hybrid environments