2023 OFA Virtual Workshop

RDMA and Linux TCP

David Ahern, Roland Dreier, Shrijeet Mukherjee

enfabrica

Why we looked at RDMA and TCP

enfabrica

- Trend towards converged ethernet networks
  - RoCE/IB has solved most problems
  - Current focus is around
    - Out of Order delivery
    - Selective acknowledgement and efficient retransmission
    - User configured Congestion Control schemes
- Mixed use traffic
  - Familiarity matters for debugging at scale
  - RoCE Vs TCP on the fabric or Sockets Vs RDMA on the host
- RoCE is still run on separate networks
  - Means storage networks also need to be converted
  - Gateway for cloud services
  - Big Data for inference needs to run on shared infrastructure (not supercomputers)

Most Important : Is there something fundamentally wrong about TCP

Take Another Look at TCP

enfabrica

- Rich Congestion Control algorithms
  - actively researched, with many behavioral models
  - highly effective mechanisms for rate control
  - selective acknowledgement and efficient retransmission model
  - supports out of order packet delivery and in order completion

- TCP re-implementations have been attempted before
  - iWarp
    - Built by mating RDMA semantics over a TCP reliable fabric
    - Fast moving TCP stack had to have hard to change hardware/firmware implementations
  - Onload ala Solarflare
    - Userspace, per process communication stack
    - Reasonable success in HFT/HPC applications
  - DPDK implementations of TCP
    - Success in middle box implementations
    - mostly used to implement low overhead packet processing on CPU's

Making it work at speed on "simple" hardware would make deployments simpler

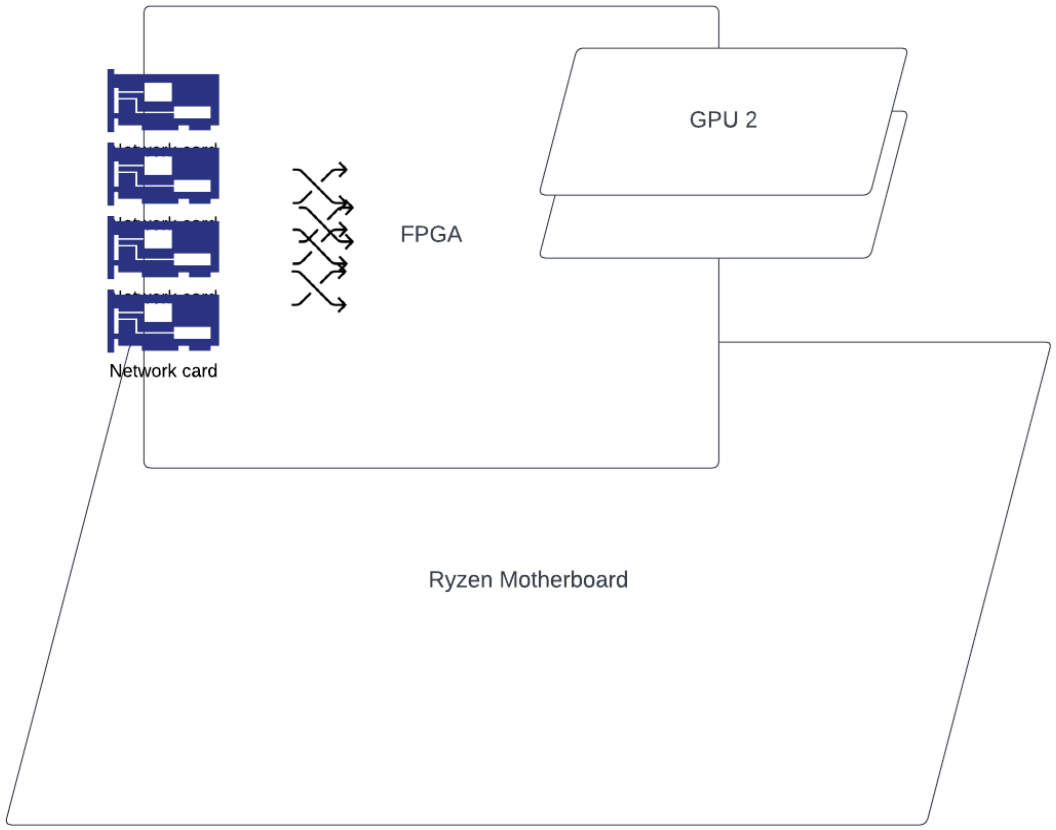
© 2023 ENFABRICA CORPORATION. ALL RIGHTS RESERVED

what holds TCP back

enfabrica

- Sockets
  - syscall overheads are significant
- No support for messaging
  - Need framing and out of order message delivery support on top of a byte stream
  - Upper level implementations exist, like NVMe-TCP, but creates PDU parsing complexity
- No zero copy
  - Current linux kernels have a good Tx zero copy implementation
    - Not as efficient as RDMA send/write
  - Rx zero copy is limited with awkward semantics
  - Does not work for heterogeneous memory e.g. GPU's
- Precise placement and management of Buffers not available to userspace applications
  - Is a function of the sockets API, not TCP
  - Needed for ML applications or userspace based storage stacks

FPGA-Based Test Platform



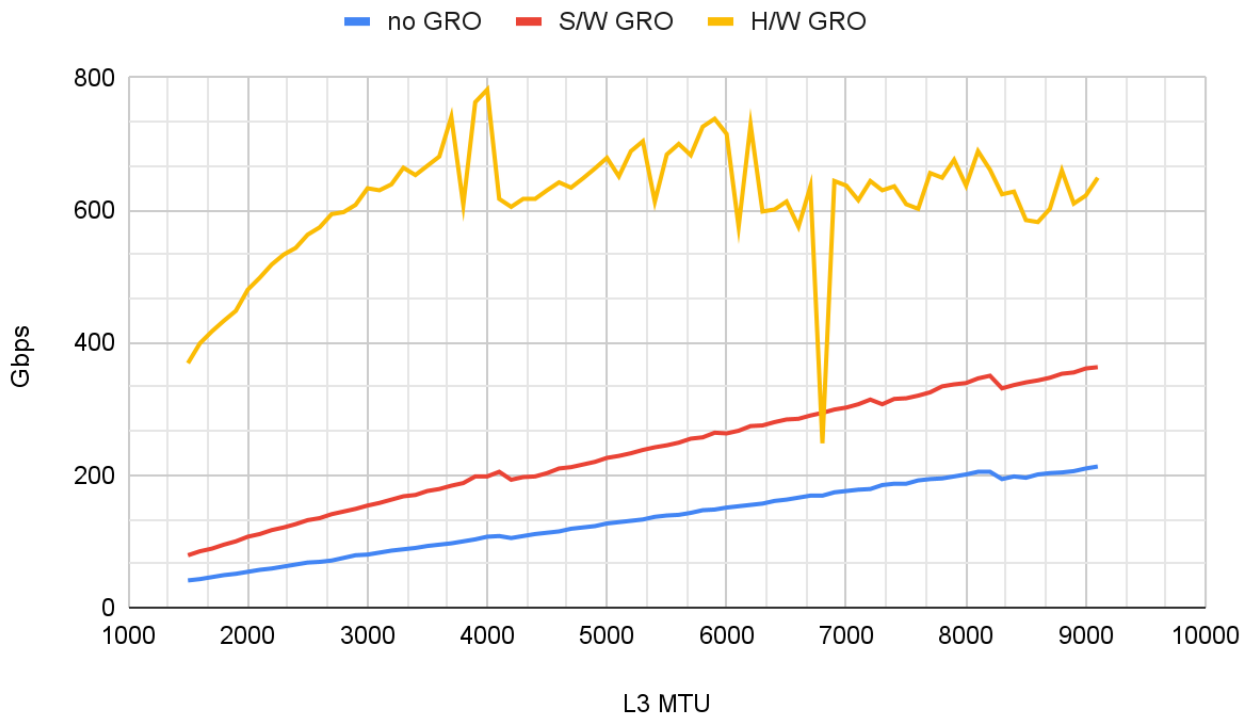
- Virtex UltraScale+ XCVU37P
- Xilinx PCIe Gen3 x16 lane (~100Gbits/s) interface to the CPU
- 2 x PCIe Gen3 x16 lanes to the GPUs, limited to ~40Gbps for TX
- 4 x 100G Ethernet CMACs with routing lookup BCAM
- Architectural approximation to our design
  - HW support for TCP classification
  - Flow offset tracking
  - TSO/GSO

Can we build onload using the stock Linux stack ?

Linux TCP/IP Stack

enfabrica

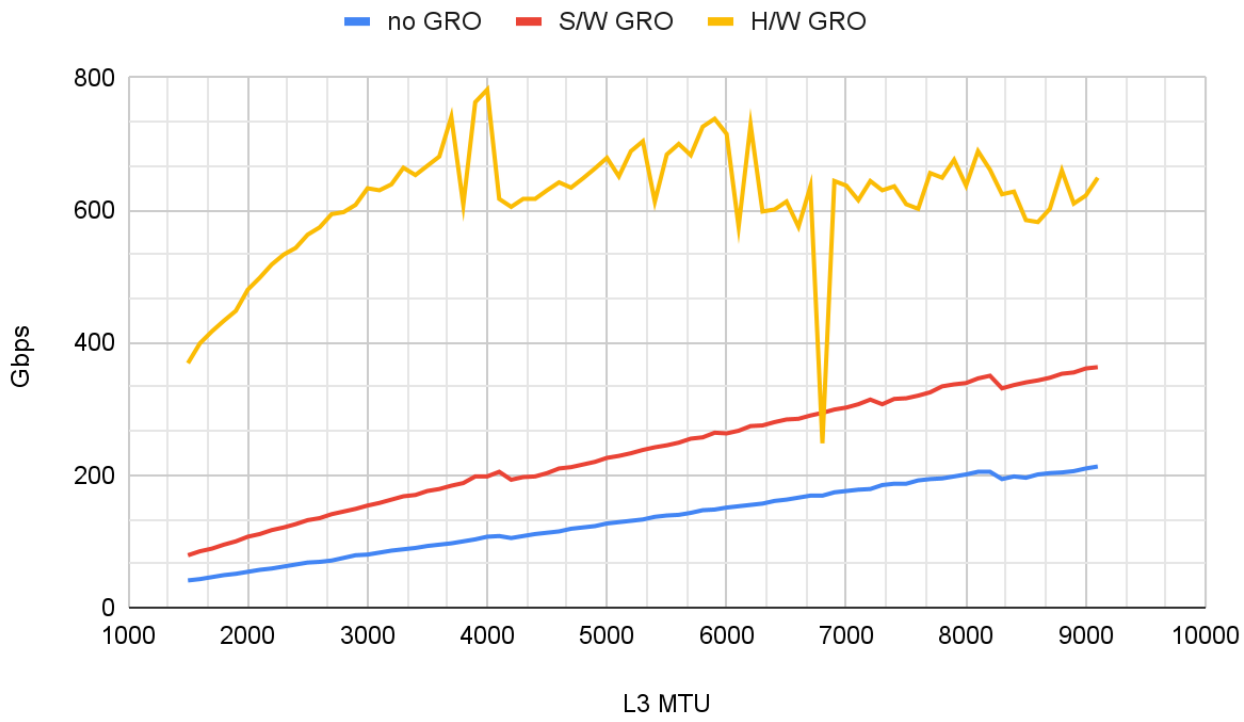
- Core of the Linux TCP/IP stack more than capable of running at high speeds
  - Syscall/Socket API's are the main issue
  - Driver level resource allocation/caching is key
  - SACK built-in
  - OOO Data and IO Completion model
- Stack is being continuously advanced and provides big gains (5-10 commits/month)
  - improvements to congestion control algorithms
  - TSQ
  - flowlet routing (check)
  - dynamic congestion control using ebpf
  - new features such as "Big TCP" (more data per S/W packet - IPv4 in v6.3 and IPv6 in 5.19)
  - …



Clearly the stack can keep up

Linux TCP/IP Stack

- CPU utilization is clearly the key problem
  - Packet rate can be maintained
  - Cannot come at cost of fixed latency
  - Needs efficient resource management
  - Needs clever management of GRO techniques
- more details here
  - *LPC-2022 HighSpeedLinuxNetworking*
  - *LPC-2022 TuningLinuxTCPForDataCenterNetworks*
- Alternate solutions evolving in linux
  - io_uring
    - Similar in structure, but zero-copy Rx is evolving
    - New API, no presence in Machine Learning ecosystem
  - af_xdp
    - Good for packet processing
    - Needs a new protocol stack and API



Design : onload using the kernel stack

enfabrica

Pull in Memory Region concept

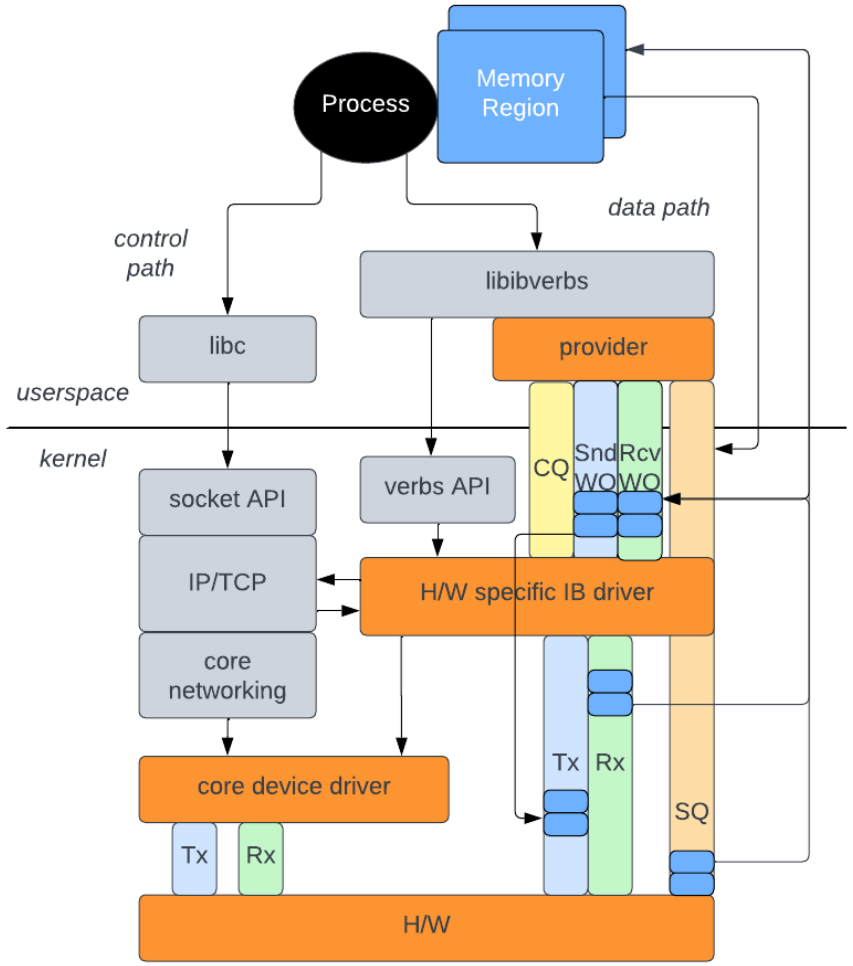
- Page pinning, DMA mapping with hardware in control path
- Reduces overhead managing buffers for connection
- Application buffer control

User-kernel S/W queues

- Reduce system calls submitting work requests and getting com
- Hardware assist for polling overhead

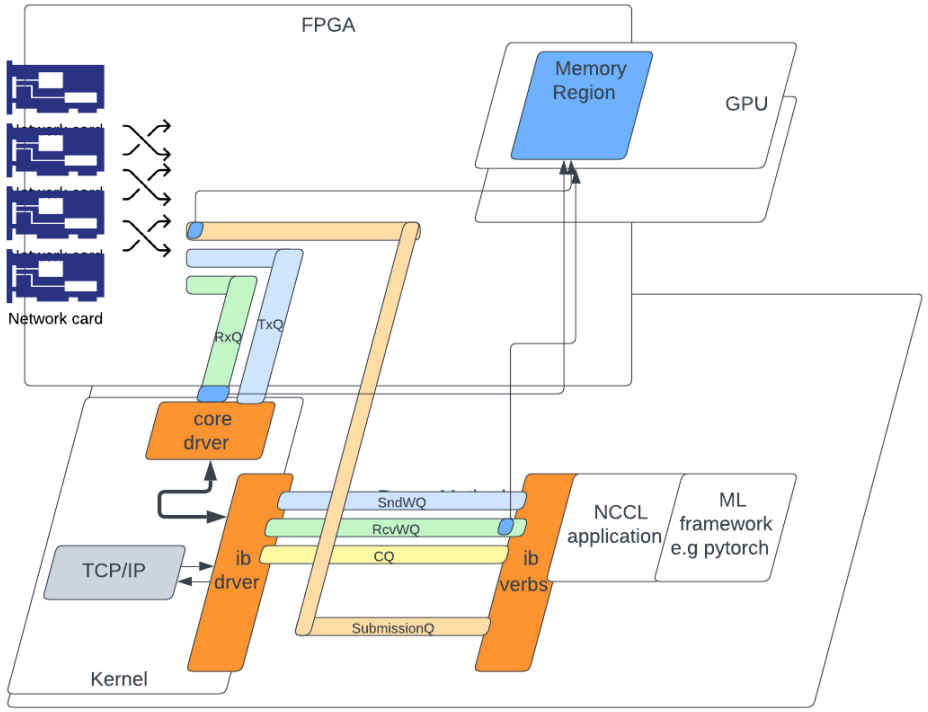
Dedicated H/W queues for flow

- Tx/Rx with application based buffers - avoids memcpy
  - Hardware tracks flow byte offsets
- RSS/flow_steering to direct packets for flow to Rx queue
- Direct from Userspace buffer submission queue



Design : Merging Networking Worlds

- Kernel module binds QP to socket
  - Manages process specific H/W queues for QP
  - Manages Rx queue and sending data through TCP
  - Manages work requests and TCP callbacks
  - Sends CQEs as Rx and Tx work requests are completed
- Provider library manages the Submission Queue and work request queue
- Data path bypasses all of the Linux infrastructure hooks
  - Those features can be enabled at performance cost
- Userspace provider and kernel module from hardware vendor allow for better hardware integration
- ALL code components exist today, the wiring is the only change
  - Same model can run RXE instead of TCP/IP



Design : TCP and messages

Solution is needed for existing message oriented applications

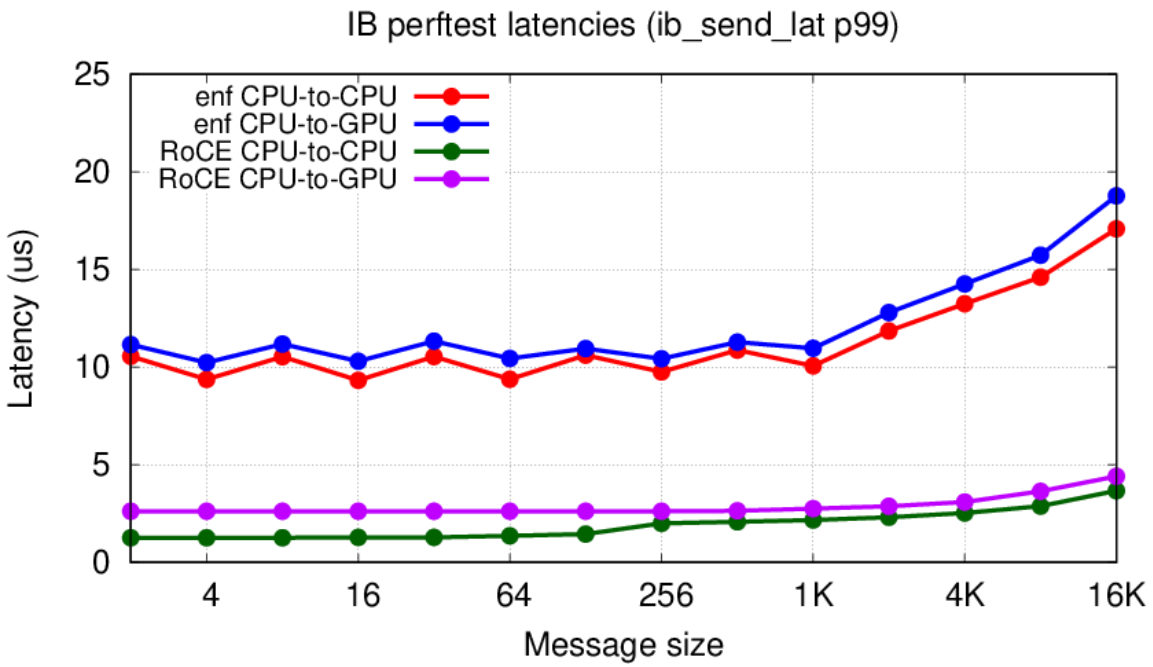
- iWARP is an existing protocol stack for RDMA layered on TCP
  - Multiple layers over TCP:
    - RDMAP (RDMA protocol) over DDP (direct data placement) over MPA (PDU markers)
  - L5 protocol
    - protocol engine must look inside full L4 payloads
    - L4 stream correctness needs to be reframed into messages

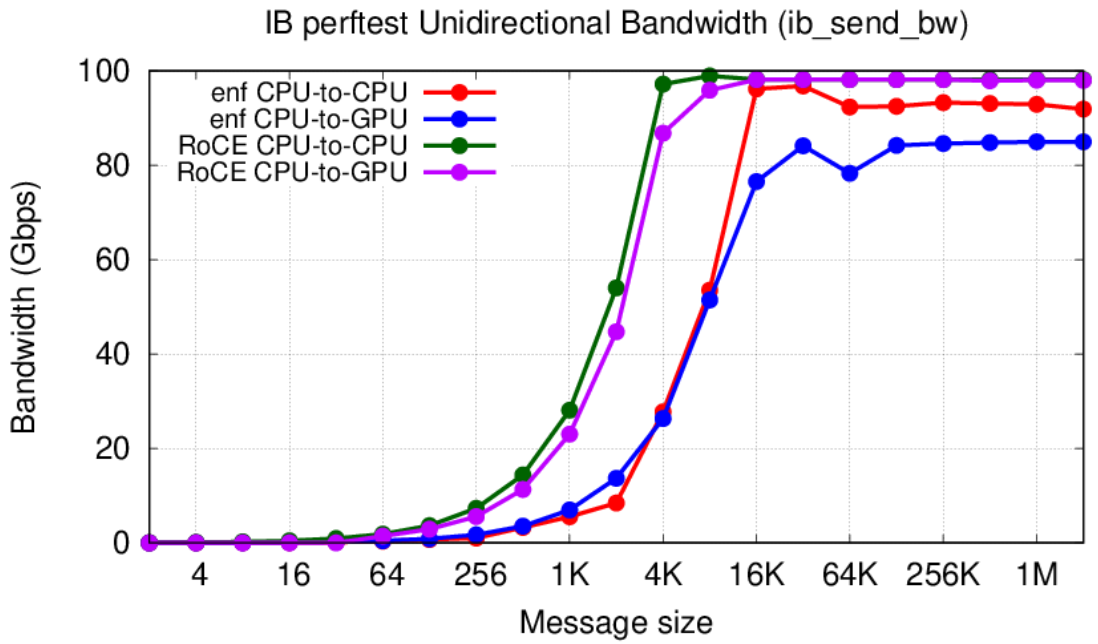
To keep hardware simple and avoid the iWarp outcome

- Control packet framing, use a “BTH-like” informational header
  - RDMA position header to create associated with TCP header
  - Expressed as a TCP option header
    - has complications due to limited size
  - Transparent to payload movement
- Tradeoff:
  - Protocol can be accelerated efficiently without stateful hardware
  - Incompatible with middle boxes that reframe TCP stream
    - (not a huge impact for clusters using 800G Ethernet)

```
u8     opcode;      /* RoCE opcode */
__be16 msn;         /* message seq number */
__be32 mbo:24,      /* message byte offset */
```

Performance Results - 100G Setup





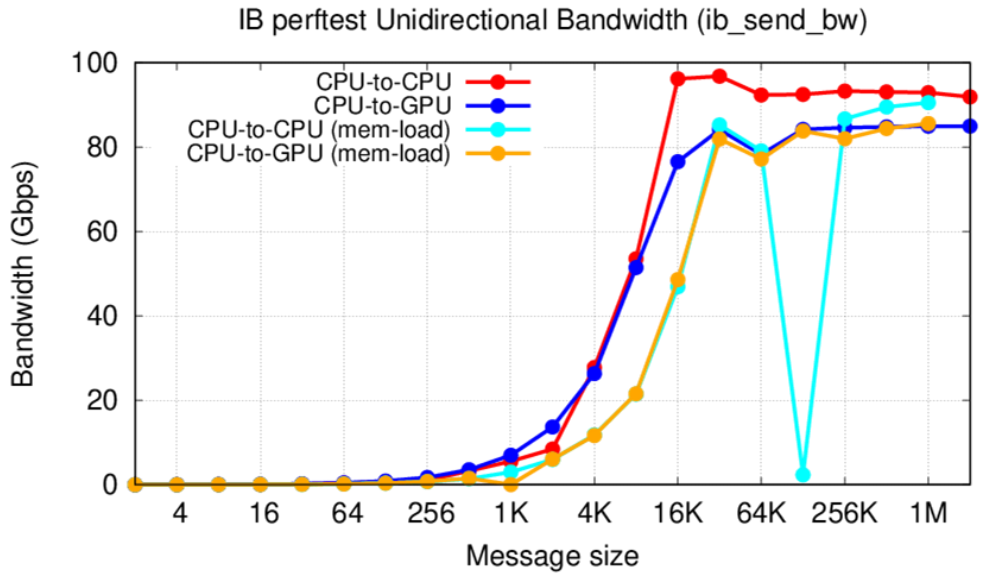
Baseline TCP latency w/ FPGA platform:

- 116us @ 4K MTU
- 100G, ~3Mpps @ 4K MTU

FPGA setup has major limitations, but proves the architectural concept

Throughput, interference and onload

- CPU based protocols are very susceptible to to interference
- Isolation of memory accesses reduces susceptibility and improves tail swings
  - Being able to place the Header and Memory region on different memory controllers has value



Impact on throughput with a memory interference pattern running concurrently

Summary

enfabrica

Keep the socket APIs for the control path and Linux IP/TCP stack

Get the OS out of the way in the data path

- Avoids system calls and memory copy
- Avoids infrastructure hooks up and down stack

Congestion control is hard and ever changing

- Build simple models that evolve at software speeds
- Benefit from years and diverse experiences of TCP
  - For many use cases the issues are not fundamental to TCP

Call to Action

enfabrica

1. Participate in the TCP stack inclusion into the RDMA ecosystem
   If RDMA and TCP were considered mutually exclusive, maybe look again

1. Participate in the Message Framing exercise for TCP

LinkedIn -> https://www.linkedin.com/company/acebinfra1/
Tech Blog -> https://blog.enfabrica.net/

Questions, Thoughts, Patches shoot us email at hello@enfabrica.net

Thank You

Questions, comments, or collaboration :

shrijeet@enfabrica.net
david@enfabrica.net
roland@enfabrica.net