RoGUE: RDMA over Generic Unconverged Ethernet

Yanfang Le

with Brent Stephens, Arjun Singhvi, Aditya Akella, Mike Swift
RDMA Overview

- **RoCE**: a protocol that provides RDMA over a lossless Ethernet network

Low Latency, High throughput, Low CPU utilization
RoCE assumes Ethernet network to be lossless – achieved by enabling Priority Flow Control (PFC).
Motivation

• Data center providers are reluctant to enable PFC
  – Instead, isolate RDMA traffic and TCP traffic

• RDMA has not seen the uptake it deserves
Can we run RDMA over generic Ethernet network without any reliance on PFC?
Can we run RDMA over generic Ethernet network without any reliance on PFC?

RoCE + PFC
Congestion Control
No packet drop

RoGUE
Congestion Control
Retransmission
yet retain low latency, CPU utilization
RoCE Overview

- Verb
- RDMA APP
- Send QUEUE
- Receive QUEUE
- Completion QUEUE
- CPU
- Signal
- RNIC

Brake the animations
## Where to fix: HW or SW?

**Hardware**
- Low CPU utilization, Low Latency
- It requires to work with NIC vendor
- Heterogeneous network hardware with non-standard protocol implementation
- Complicates network evolution

**Software**
- Easy to implement
- Packet level congestion signals are unavailable
- High CPU utilization if per-packet operations
RoGUE Overview

- Congestion Control
  - Congestion Control loop
  - CPU-efficient segmenting
- Loss Recovery
  - Shadow Queue Pair
- CPU
- RNIC
  - Hardware timestamp to measure RTT
  - Hardware retransmission
  - Hardware rate limiter to pace packets
Congestion Signal

- RTT is high, the queue builds up, reduce the sending rate
- RTT is low, network is idle, increase the sending rate
CPU Efficient Segmenting

• Two key questions
  • How large a verb should RoGUE send?
  • How often should the RNIC signaled?

• Small Verb (< 64KB)
  • signal every 64KB
  • CPU utilization (< 20%)

• Large Verb (>= 64KB)
  • chunk, and signal every 64KB.
  • CPU utilization (< 10%)
RTT measurement

$T_{\text{enc}_s1}$

$T_{\text{enc}_s2}$

$T_{\text{start}_s2}$

$T_{\text{comp}_s1}$

$T_{\text{comp}_s2}$

$T_{\text{start}_si} = \max($ Verb i enqueued, last packet of Verb i-1 goes out of NIC $)$

$\text{RTT}_i = T_{\text{comp}_si} - T_{\text{start}_si} - \text{bytes}/\text{rate\_limit}$

RTT is measured by Hardware timestamp.
Congestion Response

• Similar to TCP Vegas, and Timely
• If congestion window >= 64KB, window-based + rate limiter
• If congestion window < 64KB, rate limiter only
• Rate limiter is offloaded to RNIC
Evaluation

• Mellanox ConnectX-3 Pro 10Gbps RNICs, DCQCN

• Baselines: DCTCP, DCQCN
Evaluation-Cluster Experiments

- Each of 16 hosts generates 1MB RPC for random destinations and send 1KB RPC once every ten 1MB RPC.
Evaluation-Congestion Response
Evaluation-CPU Utilization

![CPU Utilization Graph]

- DCTCP
- RoCE (READ RC)
- RoGUE (READ RC)
Summary

• It is possible to support RoCE without relying on PFC
• Judicious division of labor between SW and HW to do the congestion control and retransmission, yet retain a low CPU utilization
• RoGUE supports RC and UC transport types of CC
• Evaluation results validate that RoGUE has competitive performance with native RoCE