ADDRESSING ENDPOINT-INDUCED CONGESTION IN A SCALE-OUT ACCELERATOR DOMAIN

Timothy Chong, Venkata Krishnan
Timothy.Chong@intel.com Venkata.Krishnan@intel.com

April 11, 2023
Going beyond scale-up –
Scaling out accelerators is the next challenge

Scale up - Accelerators associated with a single processing node

Scale out - Pool of accelerators in a medium scale domain – communicating directly over a data fabric

Need software & hardware enhancements to address this challenge. This includes network protocol enhancements.
Contributions

- Hardware-based networking policy targeting endpoint host congestion
- Targets medium-scale (100’s) domains, leveraging underlying physical connectivity from the loosely coupled domains
- Works on a lossy fabric, eliminating the need for PFC (unlike RoCEv2)
- More reactive and less conservative than traditional TCP-like policies
Host Congestion: A traditionally overlooked issue

Network delays and congestions have traditionally been viewed as the primary bottleneck (RDMA [7], RoCEv2 w/ ECN & PFC, TIMELY [8])

- **Host Congestion** has become increasingly significant
- Growing Link Bandwidth Delay Product (BDP)
  - 400Gbps switches on the horizon
- Stagnant host performance improvement
  - core speeds/counts, NIC buffer sizes, etc
- Recent work has begun to focus on identifying and addressing host congestion sources [1,2]
  - IOMMU translation cost
  - LLC cache misses
  - Memory access latencies
Communication Domains

Node – tightly coupled coherent domain
- BEST connectivity – typically coherent shared memory
- Latency threshold < 2us

Somewhat tightly coupled (e.g. 1K nodes)
- BETTER connectivity > 400Gbps per node injection
- Latency threshold < 8us
- Customized transport over a standard network

Loosely coupled – data center/HPC system (e.g. 64K nodes)
- GOOD connectivity > 100Gbps per node injection
- Traditional network semantics

Small
Medium
(sweet spot for a scale-out accelerator domain)
Large
COPA background
FPGAs as autonomous entities on a system

Remote accelerator invocation - no host or NIC involvement. FPGAs can be deployed as “autonomous” nodes for function-as-a-service

Network protocol enhancements for scaling-out accelerators in a medium-sized domain – this is part of COPA

COPA packet format

Applications
RDMA API (OFI/libfabric)
COPA Reliable RDMA Transport
UDP/IP
Ethernet

COPA extensions to invoke acceleration
Acceleration invocation support integrated
Target plays an active role (receiver-driven) for congestion avoidance & mitigation. Traditional schemes are initiator driven, with receiver playing a passive role.
## Protocol Design Goals

<table>
<thead>
<tr>
<th>Minimize modification</th>
<th>Maximize throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piggybacking on existing ACK’s infrastructure (duplicate ACK’s)</td>
<td>transmit the maximum number of packets (<em>streaming</em>) in the absence of host congestion</td>
</tr>
</tbody>
</table>

### Prevent congestion
- Receiver promptly alert senders of host congestion to avoid potential packet drop (NACK)
- Signal based on receiver processing queue depth

### Fast recovery
- Timely notification to sender when endpoint congestion subsides
Reliable PUTs and GETs

**PUT**
- Request (Initiator)
  - PUT (data)
  - ACK
- Response (Target)

**GET**
- Request (Initiator)
  - GET REQUEST
  - ACK
  - GET RESPONSE
  - ACK
- Response (Initiator)
Baseline approaches

Initiator-side windowing

(e.g. Window size of 3 packets)

Window = 1

ACK - 1

Window = 2

ACK - 2

ACK - 3

Window = 3

ACK - 4

ACK - 5

NACK-5*

AIMD (TCP-like)

Window = 1

1

Window = 2

3

2

Window = 3

6

5

4

5

4

Window = 1

6

* In TCP, 3+ duplicate ACK’s are treated as NACK
Ack immediately with largest sequence number in green region when:

1. New packet arrival
2. Packet transitions the threshold from above to below

ACKs are suppressed. Duplicate ACKs are sent to tell the sender to slow down.

ACKs are sent to tell the sender that the retransmit buffer for the packet can be freed
Methodology

• Custom simulator based on BookSim
  • Implements COPA reliable transport/UDP/Ethernet
  • Models Ethernet (lossy fabric with packet drops at switches & endpoints)
  • Baseline results validated with COPA hardware implementation

• Current results
  • Unicast traffic
  • Compared against baseline and AIMD policies
  • Varying host processing bandwidth from 50-80% of link bandwidth
New scheme eliminates packet drops

Initiator-side windowing

TCP-like

New scheme

Host bandwidth drops to 80Gbps
Packet size 2048B
Queue 32KB
Quantifying packet drops across different schemes

Host bandwidth:
50Gbps

Packets drop rate normalized to host bandwidth
Queue depth normalized to bandwidth delay product (RTT * switch bandwidth)

Initiator-side windowing

AIMD/TCP-like (MSS 9K)

New scheme (MTU 9K)

Packet size

512 1024 2048 4096 8192

Higher Drop rate
Worse

Lower Drop rate
Better

Packets drop rate normalized to host bandwidth
Queue depth normalized to bandwidth delay product (RTT * switch bandwidth)
Quantifying goodput across different schemes

Host bandwidth: 50Gbps

Initiator-side windowing

AIMD/TCP-like (MSS 9K)

New scheme (MTU 9K)

Packet size

Higher throughput

Better

Higher throughput

Lower throughput

Worse

Goodput normalized to host bandwidth
Queue depth normalized to bandwidth delay product (RTT * switch bandwidth)

Shows improvement for unicast flows but savings in packet drops will improve network utilization.
Takeaways

Reduced network congestion

- Fewer packet drops and retries result in lower network traffic, reducing the likelihood of congestion

Improved effective network bandwidth

- Minimizing packet retries reduces switch bandwidth wasted due to retransmissions.

Improvement in end-to-end goodput

- Negligible with high host congestion, but there is savings in packet drops
- Significant if policy can reduce packet-loss-induced host idle time (when host congestion goes away)
FUTURE WORK

- Extend studies to multi-node flows
- Expand BookSim model to include traffic generation with workloads
- Explore various network configurations
- Integrate policy into COPA transport and implement on an FPGA