HierCC:
Hierarchical RDMA Congestion Control

Jiao Zhang, Yali Zhang, Zixuan Guan, Zirui Wan, Yinben Xia
Tian Pan, Tao Huang, Dezhi Tang, Yun Lin
Ultra-low latency is needed!

Cluster-wide storage system

- IOPS 1M
- Latency<45us

Machine-learning system

- Latency<30us

Resource disaggregation

- Latency<3~5us
Hard to balance throughput and latency

- “High throughput” expects sender to send as many packets as possible (e.g. send as line-rate)
“High throughput” expects sender to send as many packets as possible (e.g. send as line-rate)

“Ultra-low latency” expects sender to send as few packets as possible (e.g. send as min-rate)
Hard to balance throughput and latency

- “High throughput” expects sender to send as many packets as possible (e.g. send as line-rate)
- “Ultra-low latency” expects sender to send as few packets as possible (e.g. send as min-rate)

Hard to balance ultra-low latency and high throughput
Today’s solutions

Three main options

Reactive congestion control
- DCQCN(SIGCOMM’15)
- TIMELY(SIGCOMM’15)
- HPCC(SIGCOMM’19)
- Use congestion signal
- Always react after congestion happens

Proactive congestion control
- ExpressPass(SIGCOMM’17)
- NDP(SIGCOMM’17)
- Homa(SIGCOMM’18)
- Use credits to schedule packets
- Meet first RTT problem

Centralized control
- FastPass(SIGCOMM’14)
- Flowtune(NSDI’17)
- Use a central arbiter to globally schedule packets
- Need one extra RTT to request credits.
Today’s solutions

Three main options

Reactive congestion control
- Use congestion signal
- Always react after congestion happens

Proactive congestion control
- Use credits to schedule packets
- Meet first RTT problem

Centralized control
- Use a central arbiter to globally schedule packets
- Need one extra RTT to request credits.

But these solutions face a dilemma in handling traffic uncertainty!
Two common cases with uncertain traffic (Case 1)

Case 1: Short burst flows

- Using HPCC (Make control after congestion happens)
- Need multiple rounds to converge to the ideal rate
Two common cases with uncertain traffic (Case 1)

Short burst flows

- Queue length of the bottleneck link and overall throughput with 25Gbps link speed
- HPCC needs 5(#maxstage) RTT to converge to the ideal rate
Two common cases with uncertain traffic (Case 1)

Short burst flows

Existing reactive CC mechanisms:

• Most existing reactive CC mechanisms use line-rate as the initial rate.
• When short burst flows start, the existing flows that have already converged to the stable state have to converge to the desired rate again.
• When short burst flows finish, the remaining flows need multiple rounds to occupy the bandwidth released by the completed flows.
Problem 1: Perform well in stable state, but the instantaneous performance is poor with bursty traffic.

- When short burst flows finish, the remaining flows need multiple rounds to occupy the bandwidth released by the completed flows.
Case 2: Incast scenario

- Reactive congestion control can’t effectively limit the queue length when massive flows burst.
Two common cases with uncertain traffic (Case 2)

Incast scenario

- Queue length peak grows by the number of flows
Case1: Short burst flows

- When RTT grows, converging time changes from 30us to 60us, and queue length peak changes from 6MB to 9MB

Case2: Incast scenario

Long control loop makes performance worse
Case 1: Short burst flows

Case 2: Incast scenario

Problem 2: Performances become worse when RTT grows!

- When RTT grows, converging time changes from 30us to 60us, and queue length peak changes from 6MB to 9MB
What about using proactive solutions?

Also meet the first RTT problem!\textsuperscript{[1]}

\textbf{#1 Use the first RTT time to schedule?}

\begin{itemize}
  \item \textbf{1\textsuperscript{st} RTT}
    \begin{itemize}
      \item Sender
      \item Receiver
      \item credit request
      \item credit
    \end{itemize}
  \end{itemize}

\begin{itemize}
  \item \textbf{2\textsuperscript{nd} RTT}
    \begin{itemize}
      \item Sender
      \item Receiver
      \item data
    \end{itemize}
\end{itemize}

\textbf{Waste the first RTT time}

\textbf{#2 Traffic burst in first RTT?}

\begin{itemize}
  \item existing flow
  \item unscheduled
  \item scheduled
  \item Buffer overflow
  \item new flow
\end{itemize}

\textbf{Packets lost happens}

[1] Aeolus: A Building Block for Proactive Transport in Datacenters 2020 SIGCOMM
What about using proactive solutions?

Also meet the first RTT problem!\[1\]

#1 Use the first RTT time to schedule?

#2 Traffic burst in first RTT?

Problem3: Hard to **accurately allocate credits** in first RTT!

Waste the first RTT time

Packets lost happens

---

[1] Aeolus: A Building Block for Proactive Transport in Datacenters 2020 SIGCOMM
Problem: Traffic uncertainty

➢ Existing solutions face a dilemma in handling traffic uncertainty.

Main Problems:

1. Existing solutions perform well in stable state, but the instantaneous performance is poor with bursty traffic.

2. Performances become worse when RTT grows.

3. Hard to accurately allocate credits in first RTT

Basic Ideas:

➢ Aggregate flows to make a long-lived flow

The aggregated flows by destination host are tightly distributed around 1MB[1].

➢ Shorten the control loop

Directly send feedback from sender-side ToR.

➢ Using a pre-credit scheme

Set initial credits for the aggregate flows.

[1] Inside the Social Network’s (datacenter) Network 2015 SIGCOMM
HierCC:
Hierarchical RDMA Congestion Control

Aggregate flows + Credit-based mechanism + Short control loop
Layer2 (inter-rack): Aggregated flows

- Aggregate flows with the same destination host at ToR switches.

Packets with the same destination host go into the same VOQ (virtual output queue)

Layer2

VOQ1.7
VOQ1.8
H1 H2 H3 H4 H7 H8
ToR1
ToR2
ToR4
C
C
H1 H7 H2 H7 H1 H8 H3 H7 C Credit
Core1
Core2
Host
VOQ
Scheduler
WRR
Scheduler
VOQ
Scheduler
VOQ2.7
WRR
Scheduler
Layer2(inter-rack): Aggregated flows

- Aggregate flows with the same destination host at ToR switches.
- Use credit-based scheme to schedule packets between ToR switches.
Layer2(inter-rack): Aggregated flows

- Aggregate flows with the same destination host at ToR switches.
- Use credit-based scheme to schedule packets between ToR switches.
- Use pre-credit scheme to alleviate the first RTT problem.

Layer2

VOQ 1.7
VOQ 1.8

VOQ 2.7

VOQ
Scheduler
WRR
Scheduler
VOQ
Scheduler
WRR
Scheduler
VOQ
Scheduler
WRR
Scheduler

Core1
Core2

ToR1
ToR2
ToR4

H1
H2
H3
H4
H7
H8

Host
Credit

Use pre-credit scheme to alleviate the first RTT problem in proactive CC solutions
Layer2(inter-rack): Aggregated flows

Aggregating flows and using pre-credit mechanism to alleviate the first RTT problem and ensure high bandwidth utilization between ToRs.
• Directly distribute rate to each individual flows.

The bandwidth obtained by an aggregate flow will be allocated to the corresponding individual flows.
• Directly distribute rate to each individual flows.
• HierCC calculate the rate based on both the VOQ arrival rate and the VOQ length in order to limit the VOQ usage.

Layer1 (intra-rack): Short control loop

Layer1
Implementation

• The sender marks one reserved bit in the data packet header periodically. The sender-side ToR then feeds a rate allocation packet for every marked packet.

In this way, sender-side ToRs do not need to maintain a flow table to store the information of active senders.

• Currently, some switch chips like Arista Jericho and Cisco Silicon One can support 100K VOQ queues.
Evaluation Setup

- **SystemC simulations and NS3 simulations**

- **Topology**
  - Fattree with 25Gbps link-rate
  - 2 Core, 4 Aggr, 4 ToR, 64 servers (SystemC)
  - 16 Core, 20 Aggr, 20 ToR, 320 servers (NS3)

- **Traffic pattern**
  - Long-lived flow + burst flows at 1us and 3us (SystemC)
  - All to all and all to one ToR patterns with 0.8 workload WebSearch traffic (SystemC)
  - All to all Cache traffic with different workloads (NS3)
HierCC limits VOQs length and it keeps high link bandwidth utilization at the same time.
HierCC reduces the FCT of both long and short flows while also significantly reducing the queue length.
HierCC reduces the overall average FCT and the 99th percentile FCT compared to other solutions.
Problem: Existing solutions face a dilemma in handling traffic uncertainty.

HierCC:
- Aggregated flows with the same destination at ToR switches.
  - make long-lived flows against the traffic uncertainty
- Use pre-credit scheme to schedule packets at first RTT
  - address the first RTT problem
- Directly distribute rate to the host
  - Shorten the control loop
Thank you!