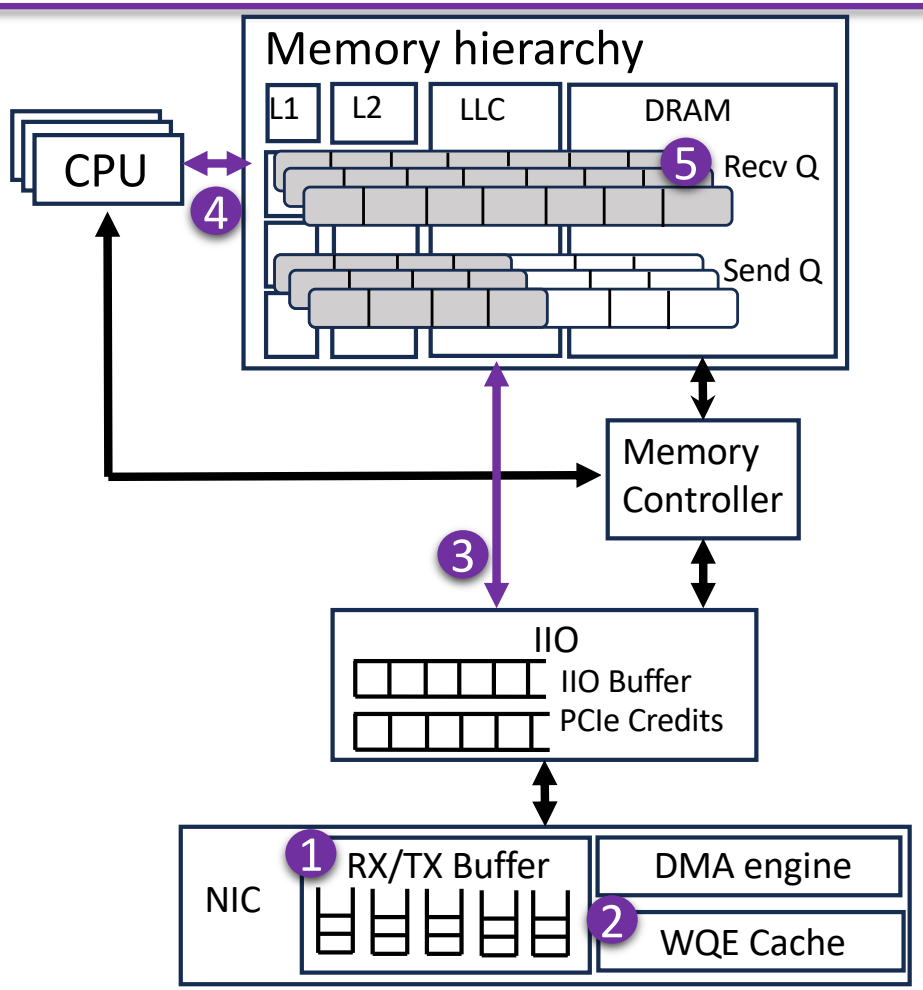
Cache-Aware I/O Rate Control for RDMA

Qijing Li, Xinyang Huang, Bowen Liu, Pengbo Li, Junxue Zhang, Kai Chen
iSing Lab@HKUST

RDMA I/O Congestion

- RDMA is a corner-stone technology in modern datacenter networks
- However, recent works revealed that congestion arises in the “last mile” of RDMA I/O path
- RDMA IO path without IO congestion:
  1. RNIC receives packets
  2. RNIC fetches WQE
  3. RNIC initiates DMA to LLC
  4. Application processes data in LLC
  5. Application replenishes WQEs

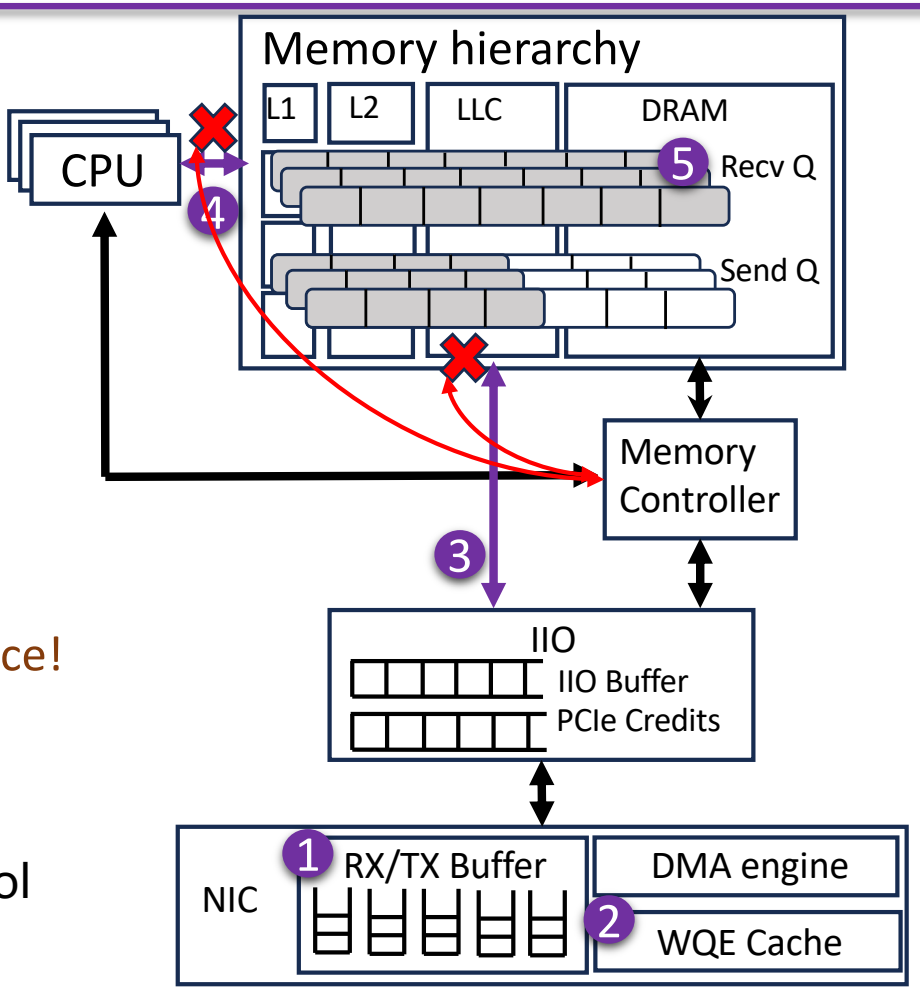


RDMA I/O Congestion

- RDMA IO path with IO congestion:
  - 3 RNIC initiates DMA to LLC
    - DDIO write allocate: allocate LLC space for new data, evict old data to DRAM
  - 4 Application ~~processes data in LLC~~
    - Application reads data from DRAM, also brings eviction

Cache eviction + Cache misses + Memory bandwidth waste → Bad Performance!

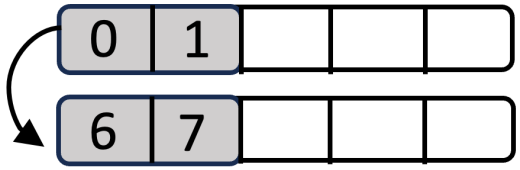
- Existing HostCC: coarse-grained rate control



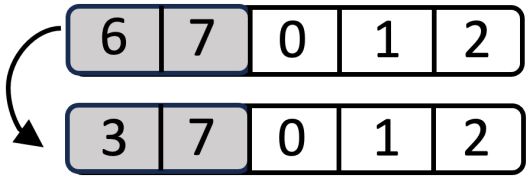
Modeling I/O Congestion

- A simple example to demonstrate cache misses in I/O path

1. RNIC DMA 8 msgs to DDIO cache
   - *evicted #0-#5 in DDIO cache*

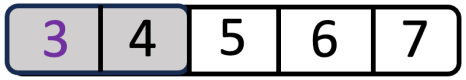


2. APP reads #0-#3 from DRAM
   - *evicted #6 in DDIO cache*



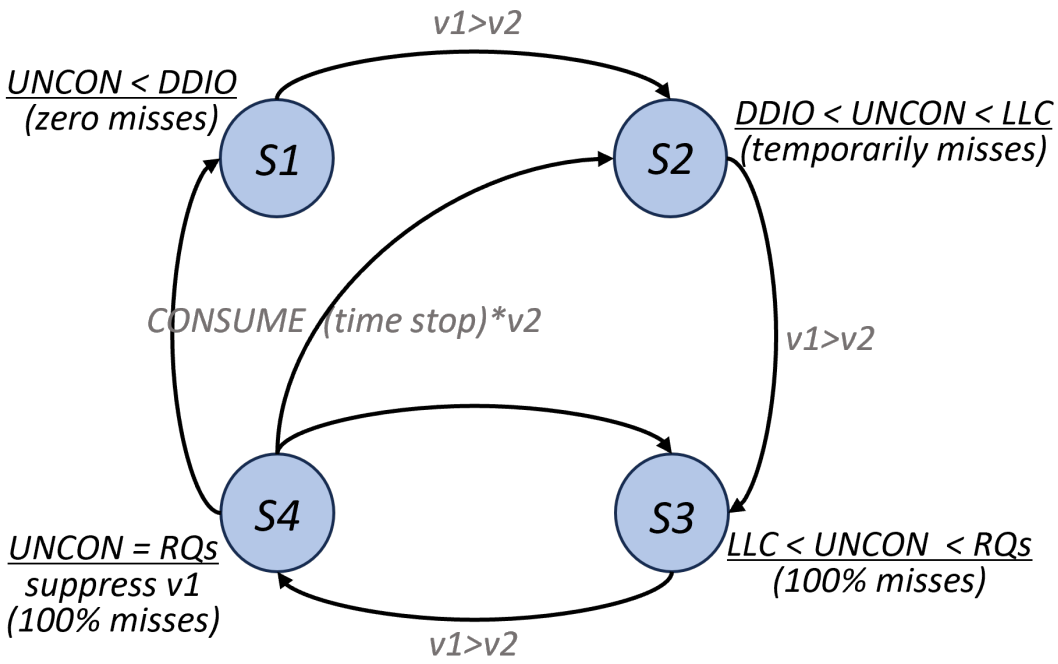
3. RNIC DMA 4 new msgs to #0-#3
   - *no eviction*

4. APP reads #4-#7 from DRAM
   - *evicted #7 , #0-#2*



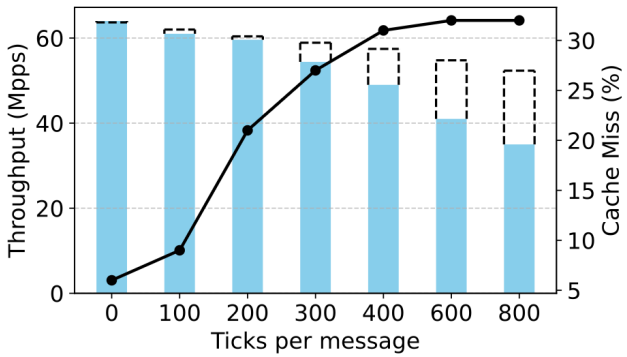
Modeling I/O Congestion

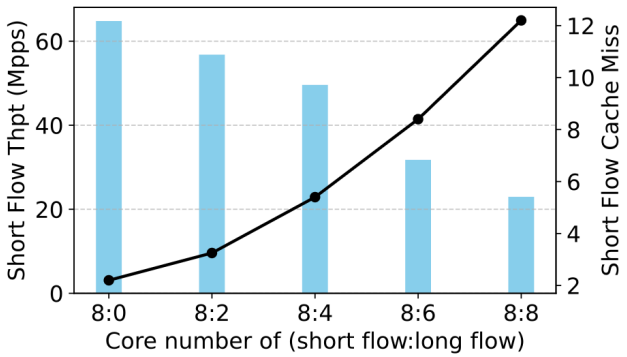
- Model I/O congestion as a producer-consumer speed mismatch problem
  - v1: network ingress rate
  - v2: CPU processing rate
  - States: represent different degree of unprocessed data volume
  - Transfer: decided by speed gap and time span
- System fluctuates among the four states
  - More time in S3 and S4 means higher I/O congestion
  - Narrow speed gap can slow down, even avoid state transfers

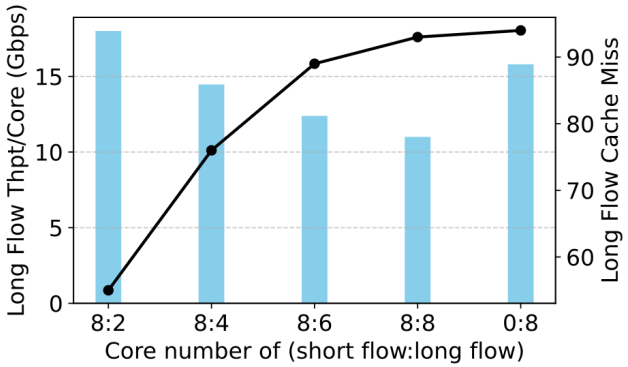


Validation of Modeling

- Choose two common scenarios of data center, increasing the gap
  - Scenario 1: single flow with different application behaviors
    - With ticks per pkt increasing, CPU processing rate decreases
  - Scenario 2: mix flows with different network behaviors
    - With number of long flows increasing, network ingress rate increases

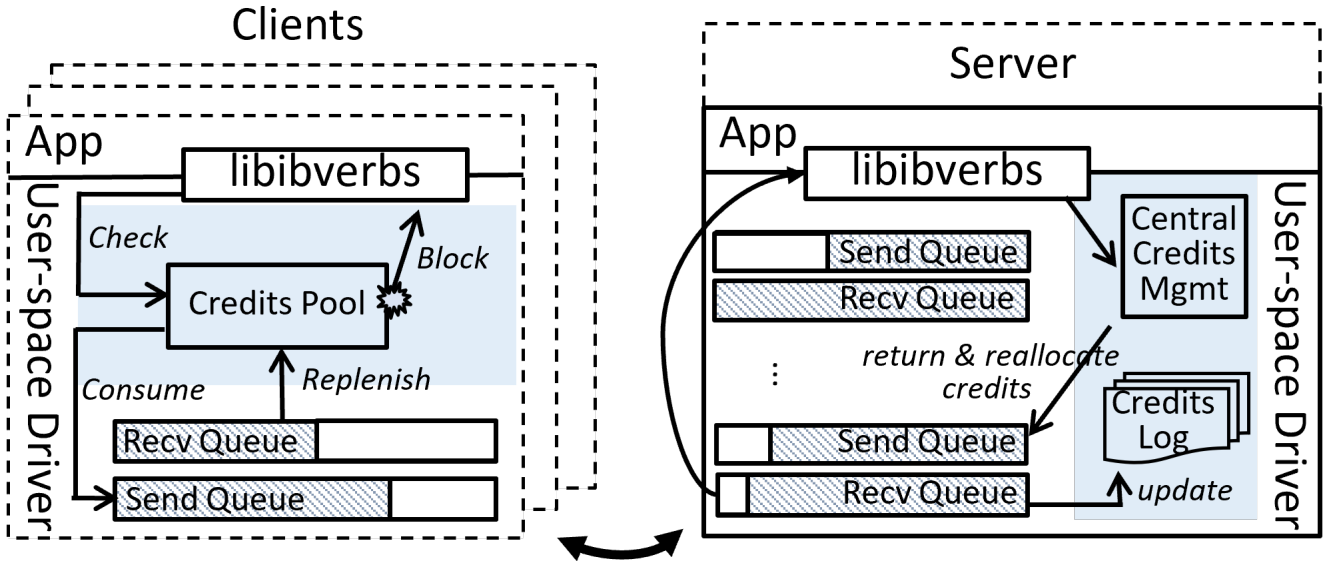






CARC Design

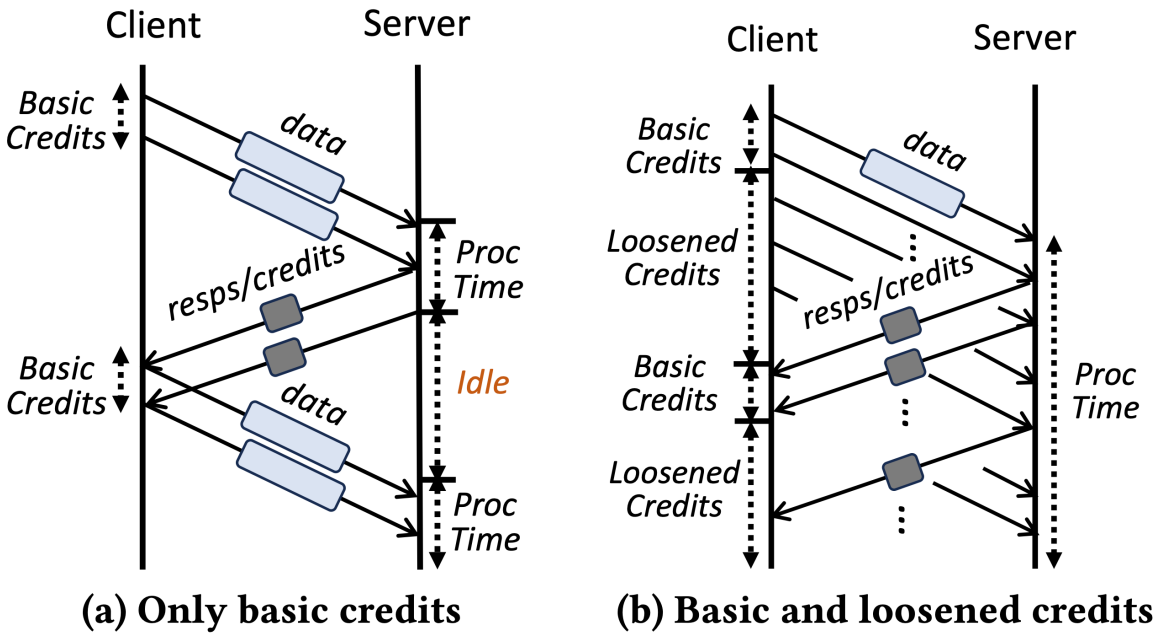
- Credits-Based Rate Control:
  - Server allocates credits upon connection establishment
  - Clients consume credits before send data
  - Server replenishes credits through piggyback or standalone signal after finishing processing data



CARC Design

- Loosened credits:
  - Adjust credits dynamically to tolerate coordination error due to RTT latency and I/O unrelated memory access

$$Credits = \frac{S_{llc}}{S_{line} \times n} + \delta$$



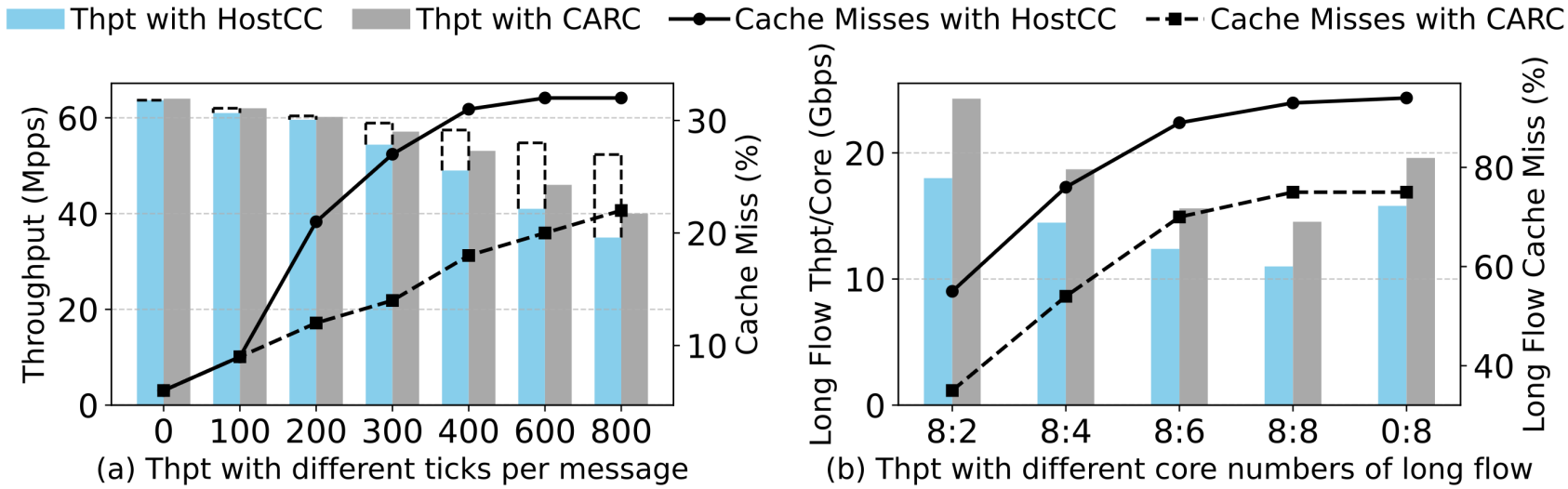
(a) Only basic credits **(b) Basic and loosened credits**

Preliminary Results

- Experiment:
  - Deploy eRPC on two 200Gbps servers and compare CARC performance with HostCC
  - CARC can reduce tail latency by up to 1.40× and improve throughput by up to 1.35× compared to HostCC.

- Vary processing speed of receiver

- Vary ingress rate of receiver



Cache-Aware I/O Rate Control for RDMA

Thank you

qlicw@connect.ust.hk