

# Optimize the TX Architecture of RDMA NIC for Performance Isolation in the Cloud Environment

**Yunkun Liao**, Jingya Wu, Wenyan Lu, Xiaowei Li, Guihai Yan  
{liaoyunkun20s, wujingya, luwenyan, lxw, yan}@ict.ac.cn

33<sup>rd</sup> Great Lakes Symposium on VLSI



# Outline

---

## ➤ Research Background

- Introduction to RDMA
- Performance Interference in Commodity RDMA NICs

## ➤ Baseline TX RDMA NIC Architecture

## ➤ Designs for RDMA Performance Isolation

- Multi-tenant RDMA Cloud Service Model
- Optimized-v1 TX RDMA NIC Architecture: Solving LS-BS Interference
- Optimized-v2 TX RDMA NIC Architecture: Solving BS-BS Interference

## ➤ Experimental Setup and Results

## ➤ Conclusions

# Introduction to RDMA

## ➤ RDMA: Remote Direct Memory Access

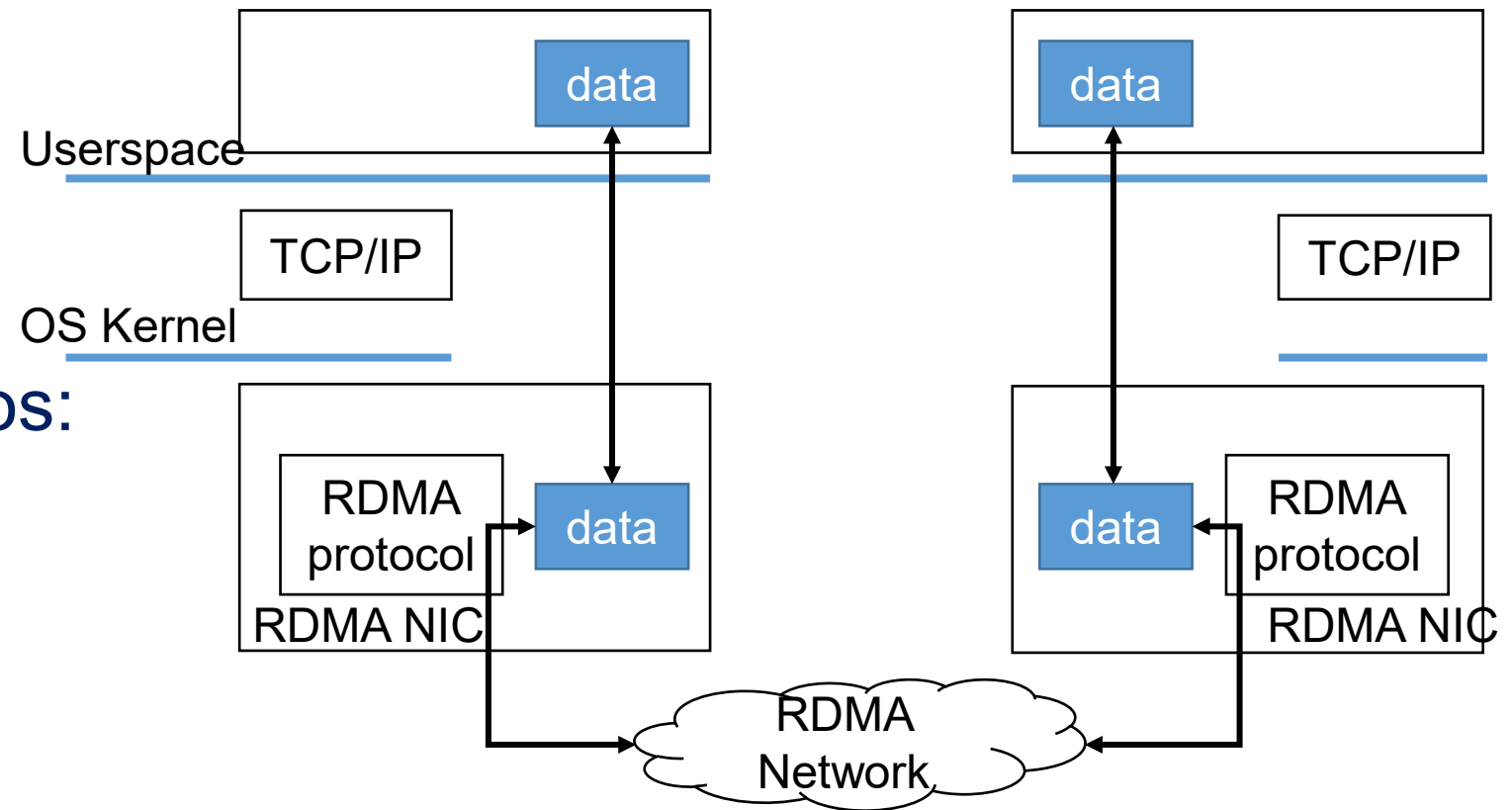
- Offloads the transport layer to the RDMA NIC (RNIC)
- Bypass the OS kernel

## ➤ Benefits:

- High throughput
- Low latency
- Low CPU overhead

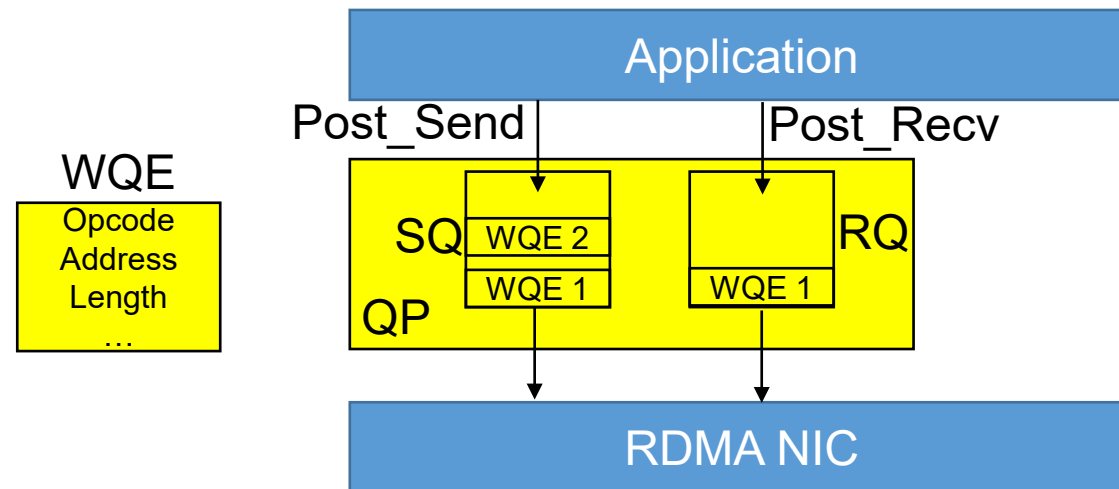
## ➤ Application Scenarios:

- Machine learning
- Web search
- Key-value store



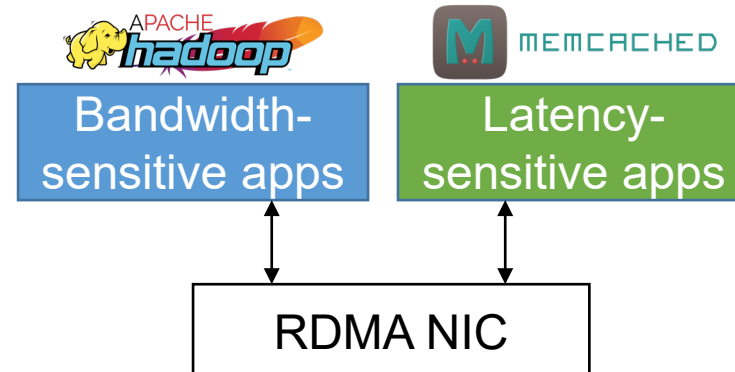
# Essential RDMA Elements

- QP: Queue Pair, interface between the application and the NIC
  - SQ: Send Queue, send message
  - RQ: Receive Queue, receive message
- WQE: basic element of the QP's SQ or RQ



# RDMA Performance Isolation is Important

- Cloud providers rely on multi-tenancy to improve RDMA NIC utilization
  - Share an RDMA NIC among multiple applications with different performance targets
- Classification of applications:
  - Bandwidth-sensitive (BS): large messages, require high bandwidth
  - Latency-sensitive (LS) : sparse small messages, require low latency
- Performance isolation: the behavior of one tenant should not affect the performance of other tenants



# Performance Interference in Commodity RDMA NICs

## ➤ Experiments setup

- 2 \* Mellanox CX-5 RDMA NICs



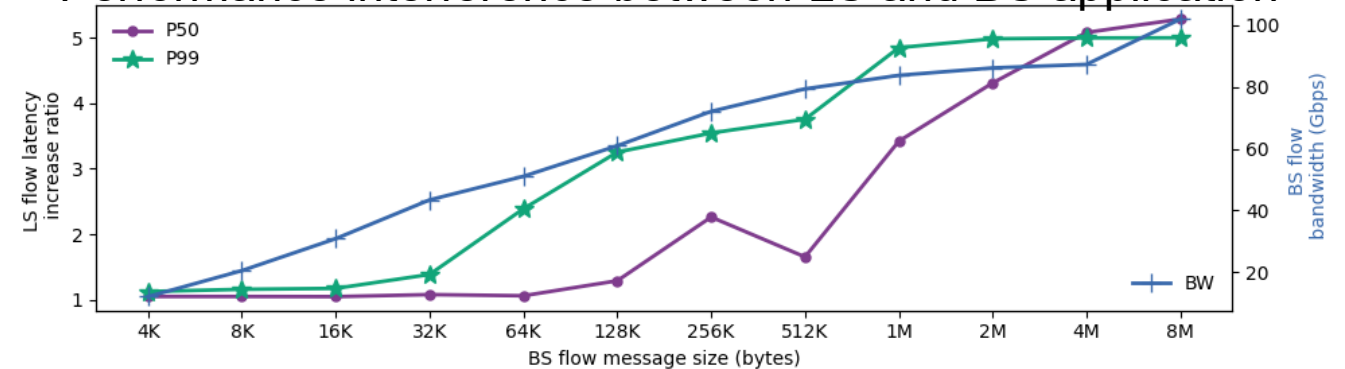
## ➤ Flow specification

- LS flow: message size = 64B
- BS flow: message size  $\geq 4\text{KB}$

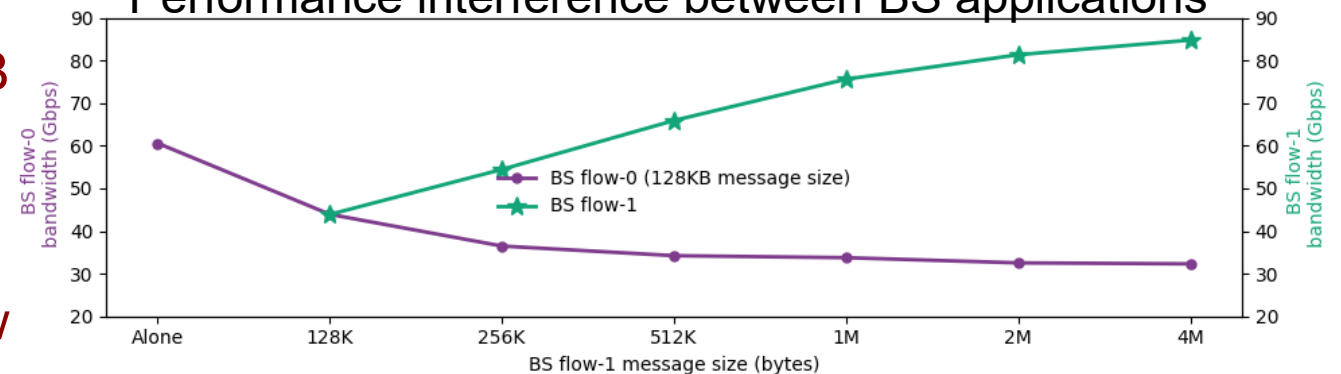
## ➤ Benchmarks

- An LS flow with a BS flow
- A BS flow with another BS flow

Performance interference between LS and BS application



Performance interference between BS applications





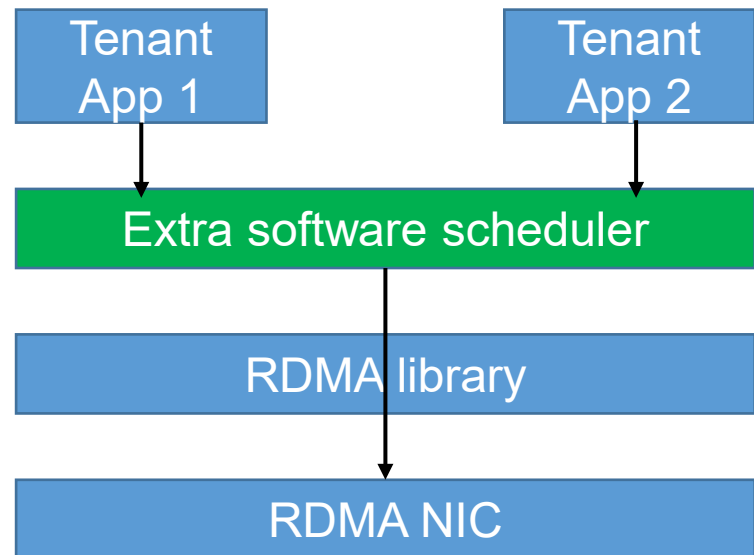
# Software Scheduling Limitations

## ➤ Performance overhead

- Extra software scheduler

## ➤ CPU overhead

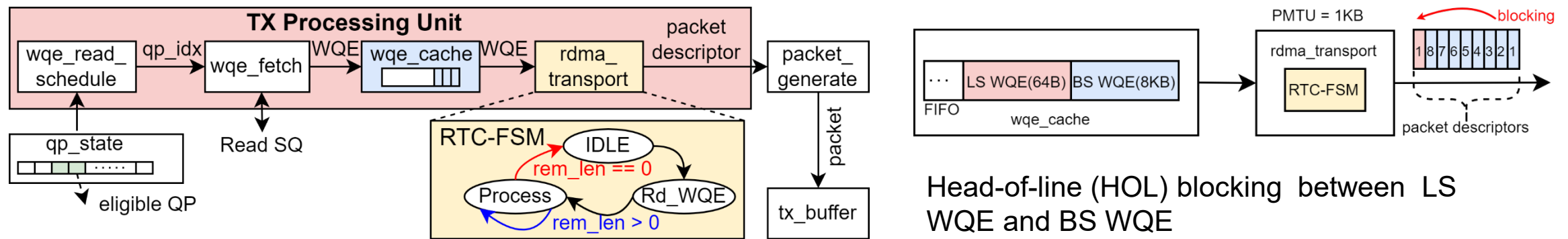
- Running software scheduler consumes CPU.



**Our work aims to achieve performance isolation from optimizing the transmit-side (TX) RDMA NIC architecture.**

# Why performance interference arise

- Challenge: commercial RNIC is a secretive 'black box';
- Solution: establish a baseline TX RDMA NIC architecture;
- Root causes of performance interference:
  - Wqe\_cache module: shared FIFO queue for prefetched LS and BS WQEs
  - Rdma\_transport module: Run-to-completion (RTC) scheduling processes a WQE until all the packet descriptors of the WQE output.

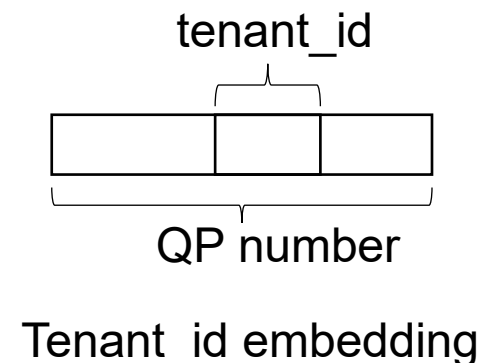
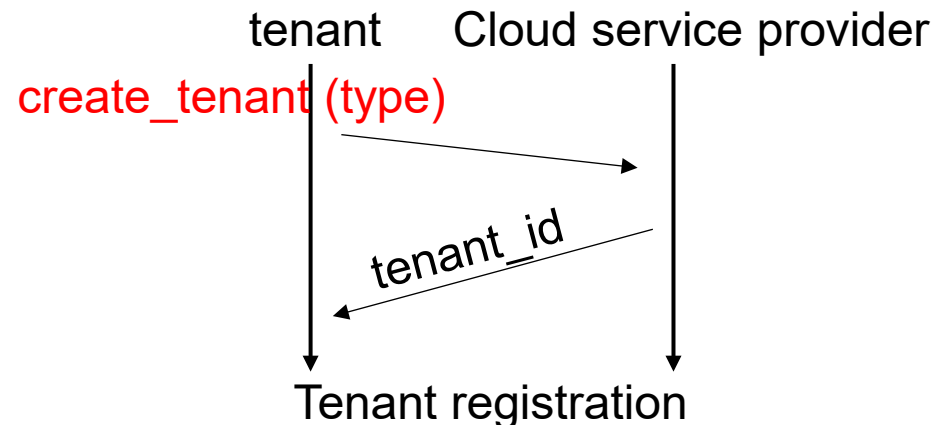


Baseline TX RDMA NIC architecture



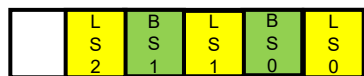
# Multi-tenant RDMA Cloud Service Model

- Problem: RDMA NIC requires application-level information from the cloud service provider.
- Assumption: the cloud service providers and the RDMA NIC vendors collaborate.
- We propose an RDMA cloud service model at the application level.
  - Embed the tenant classification into the `tenant_id`.
  - Embed the `tenant_id` into the QP number.

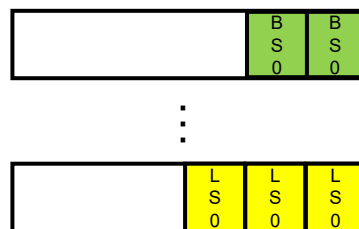


# Optimized-v1: Separate Caching

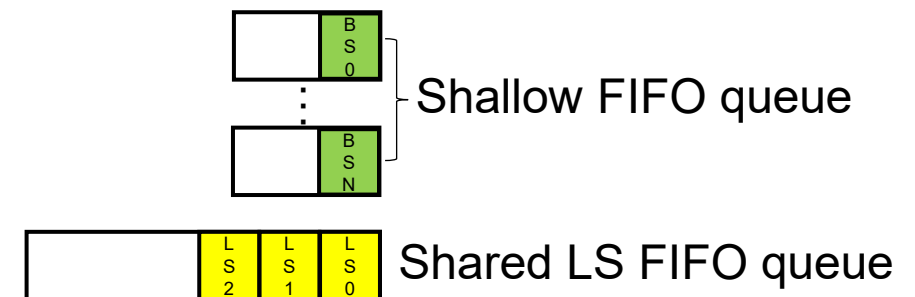
- We need to store the WQEs of LS and BS separately if we want to apply further scheduling optimization.
- Challenge: large FIFO cost in naïve separate caching
- Observation:
  - Wqe\_cache is designed for prefetching WQE and masking the PCIe latency;
  - The WQE consumption rates of LS and BS tenants are different.
- Solutions:
  - Store the BS WQEs in separate **shallow (e.g. depth =2 )** FIFO queue.
  - Store the LS WQEs in a shared FIFO queue



Baseline shared caching



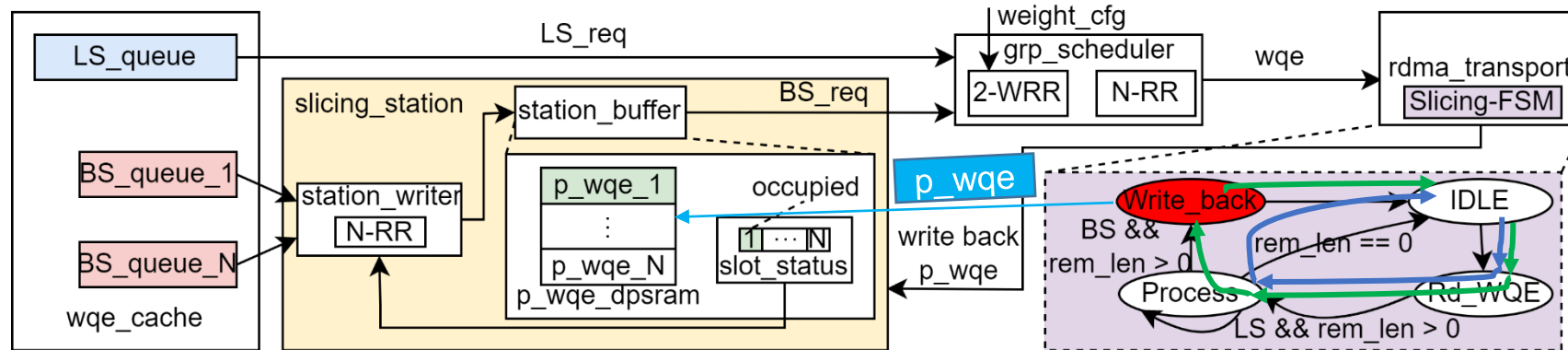
Naïve separate caching



Our separate caching

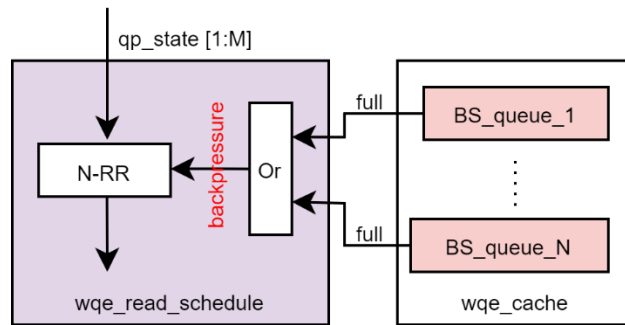
# Optimized-v1: Slicing Execution

- Slicing execution: makes BS tenants actively yield the right to use rdma\_transport, thus eliminating HOL blocking in baseline RTC-FSM.
- Slicing\_station: store the unfinished WQEs of BS tenants (defined as pending WQE, p\_wqe) for subsequent rescheduling.
- Slicing-FSM: introduces Write\_back state to enable BS WQE to give up opportunities, thus avoiding the BS WQE blocking LS WQE.

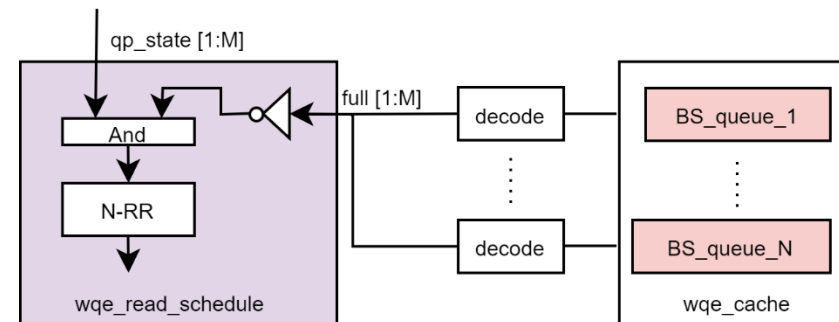


# Optimized-v2: Isolated Backpressure for BS Tenants

- Baseline coupled backpressure: any full queue will prevent `wqe_read_schedule` from scheduling.
- Problem: there is performance interference in `wqe_read_schedule` because different BS tenant has different WQE consumption rate.
- Isolated backpressure: A BS QP is eligible only when its state is eligible, and its corresponding BS\_queue is not full.



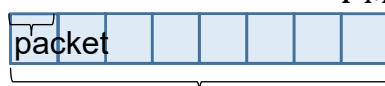
Coupled backpressure



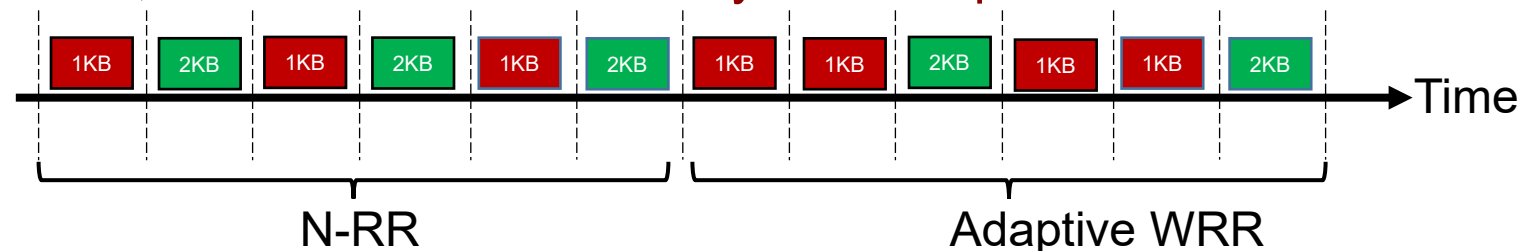
Isolated backpressure

# Optimized-v2: Adaptive WRR for BS Tenants

- Message is segmented into packets according to PMTU
- Problem: BS tenant with a larger PMTU value can get more bandwidth.
- Observations:
  - Limited distribution of available PMTUs (256B, 512B, 1KB, 2KB, 4KB);
  - The average message size of BS applications is over 1MB.
- Solution: adaptive Weighted Round-robin (WRR) to schedule multiple BS tenants instead of RR.
  - For example, if the PMTU of tenant-A and tenant-B are 1KB and 2KB, then their weight are 4 and 2. Then, tenant-A is twice as likely to be dispatched as B.

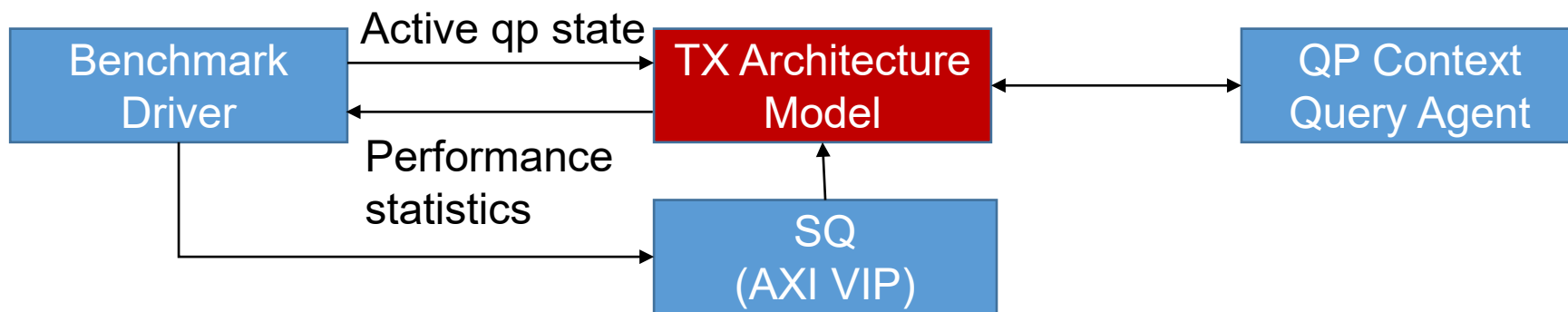
$$\# \text{ of packets} = \frac{\text{Message Size}}{\text{PMTU}}$$


message



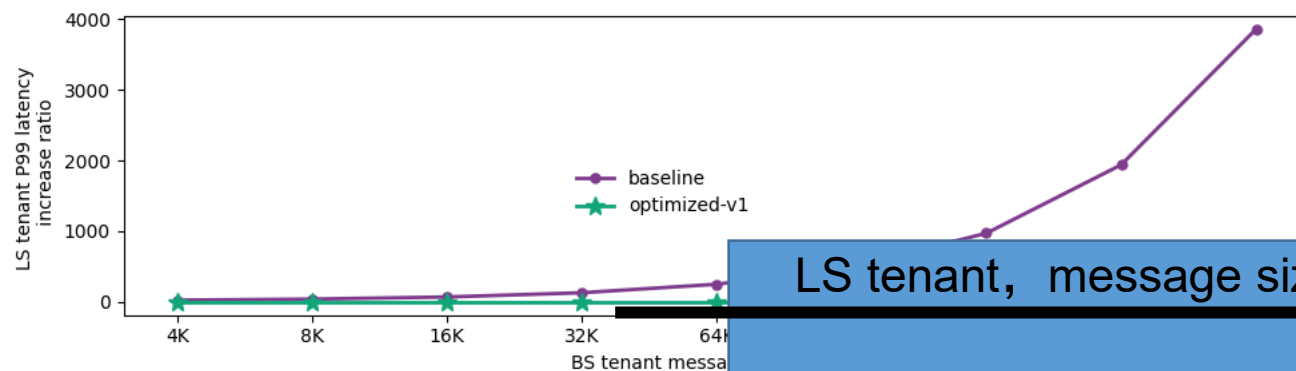
# Experimental Setup

- We implemented the baseline, optimized-v1, and optimized-v2 architectures in Verilog-HDL for circuit-level evaluation.
- These architectures are simulated and synthesized these two architectures in Vivado 22.01.
- Simulation platform:

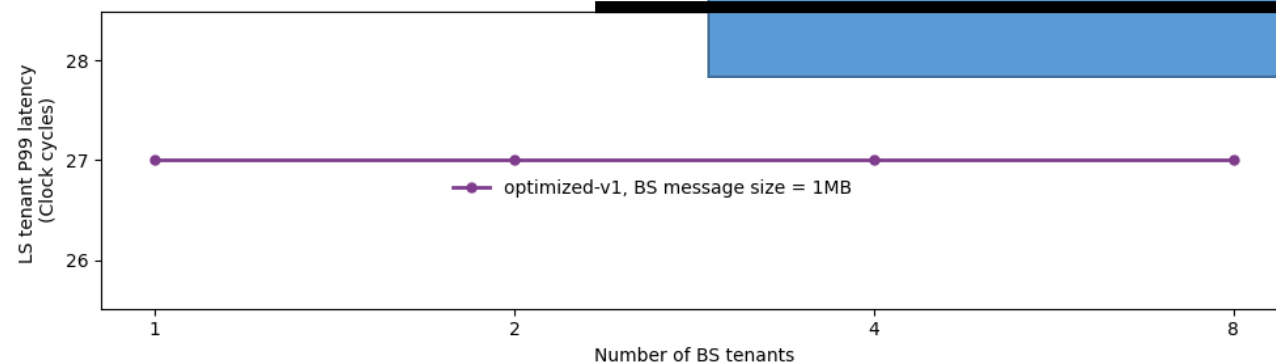




# Experimental Results: LS-BS Isolation



the optimized-v1 can nearly prevent the latency of the LS tenant

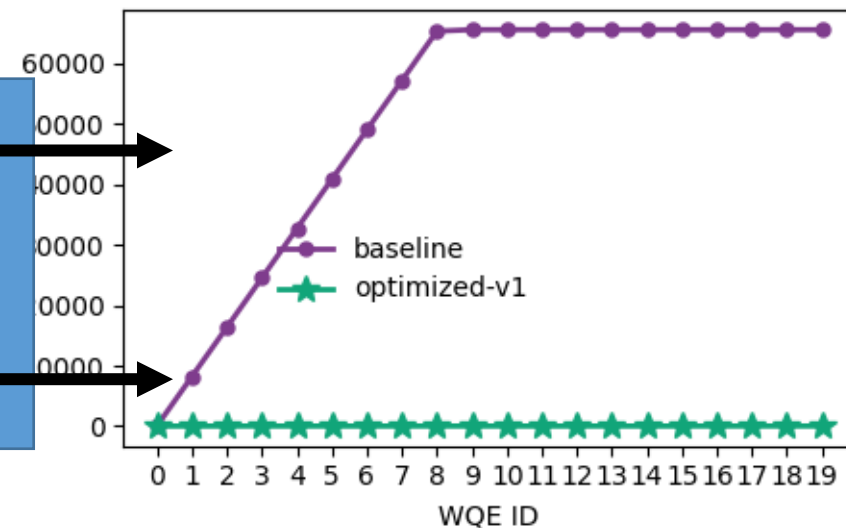


the optimized-v1 can well isolate LS and BS tenants irrespective of the number of BS tenants and the message size of BS tenants.

LS tenant, message size = 64B

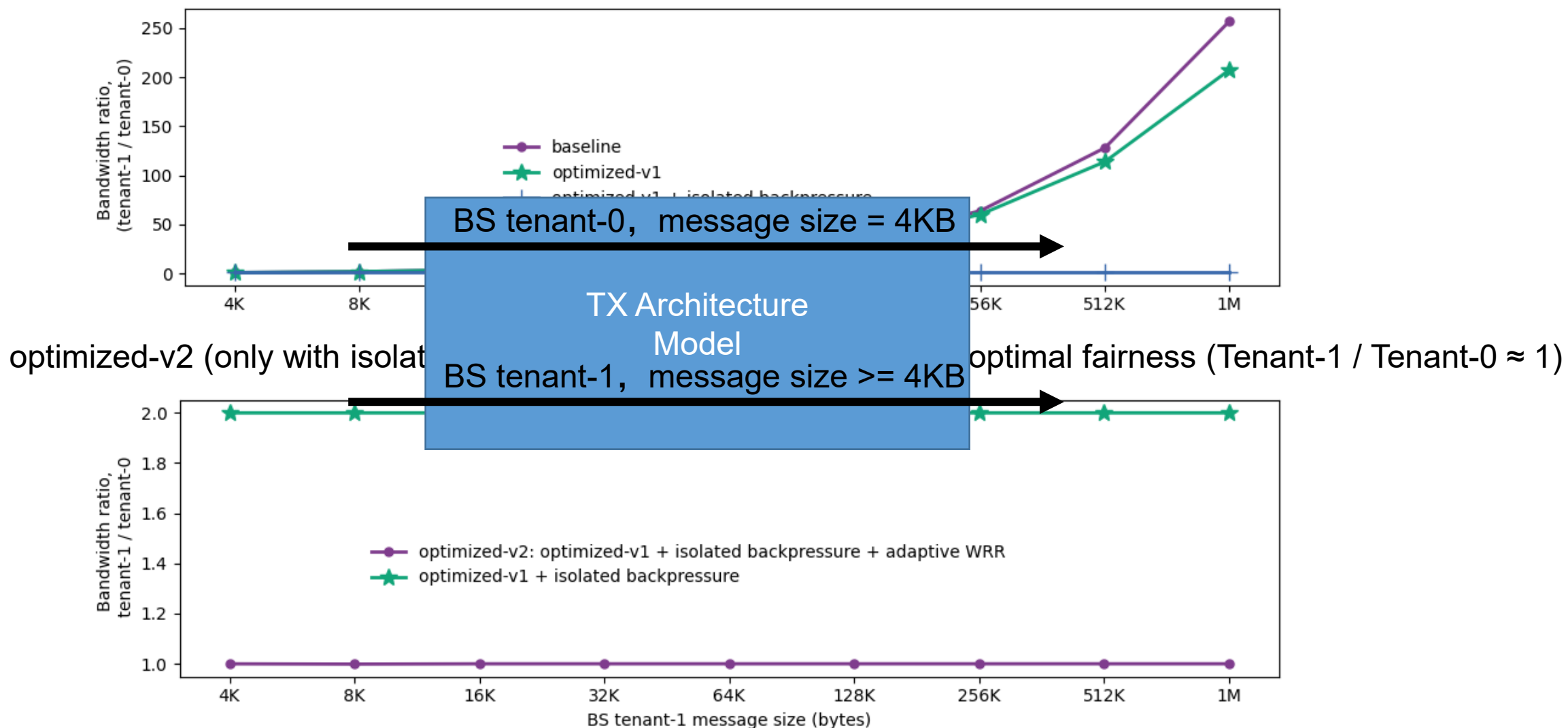
TX Architecture Model

BS tenant, message size  $\geq$  4KB



the optimized-v1 can decrease the waiting latency to about 16 clock cycles, representing one iteration of rdma\_transport.

# Experimental Results: LS-BS Isolation



The adaptive WRR can solve the unfairness caused by packet size

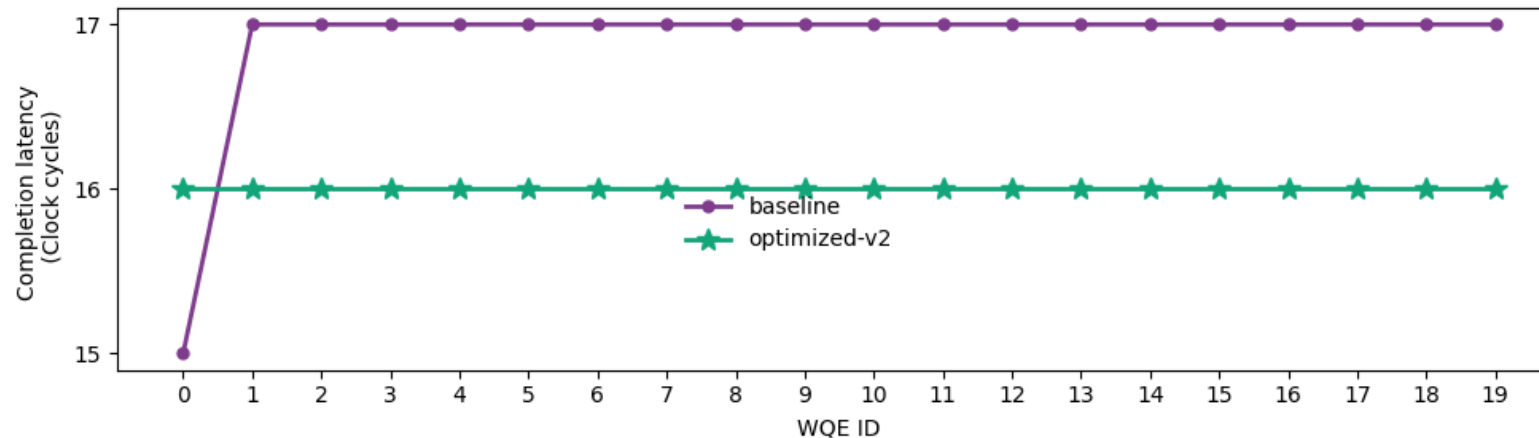
# Experimental Results: Overheads

- Area overhead: no more than 1% for CLB LUT and CLB Register when there are four BS tenants.

	CLB LUTs	CLB Registers	Block RAM
baseline	1568 (0.68%)	3035 (0.66%)	0
optimized-v1	2209 (0.96%)	5393 (1.17%)	14.5 (4.6%)
optimized-v2	3417 (1.48%)	7509 (1.62%)	14.5 (4.6%)

- Performance overhead:

- Will the latency increase for a single running latency-sensitive tenant? No
- Can the bandwidth-sensitive tenant still achieve a 100Gbps line rate? Yes





# Conclusion

---

- A baseline TX RNIC architecture to explain the performance interference of current RNICs.
- A new RDMA cloud service model.
- The optimized-v1 and optimized-v2 architecture, avoiding BS tenant's interference with LS tenants and interference between BS tenants.
- Achieving optimal isolation compared with the baseline.