PeRF

: Preemption-enabled RDMA Framework

Sugi Lee[1], **Mingyu Choi**[1], Ikjun Yeom[2], Younghoon Kim[2]

Acryl Inc[1], Sungkyunkwan University[2]

USENIX ATC '24

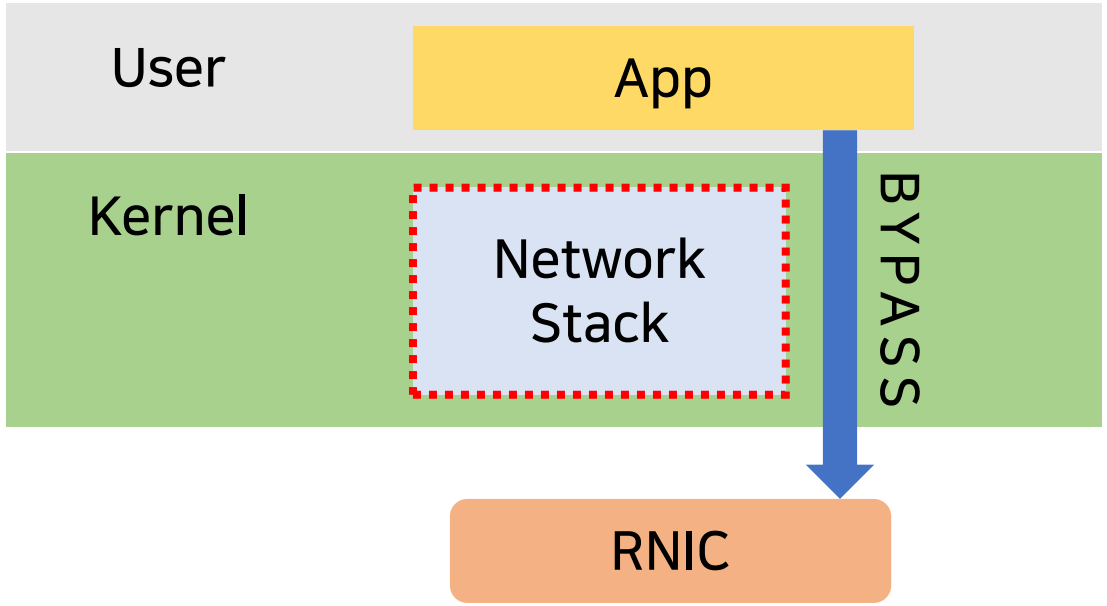
ACRYL

SUNG KYUN KWAN UNIVERSITY

RDMA

- High throughput & ultra-low latency via zero-copy operations
- Big-data analysis, machine learning, distributed storage, etc.

➔ Thanks to the kernel-bypass feature



At the same time…

- Problems such as security, scalability, and **performance isolation**

➔ Due to the lack of control in kernel space (kernel-bypass)

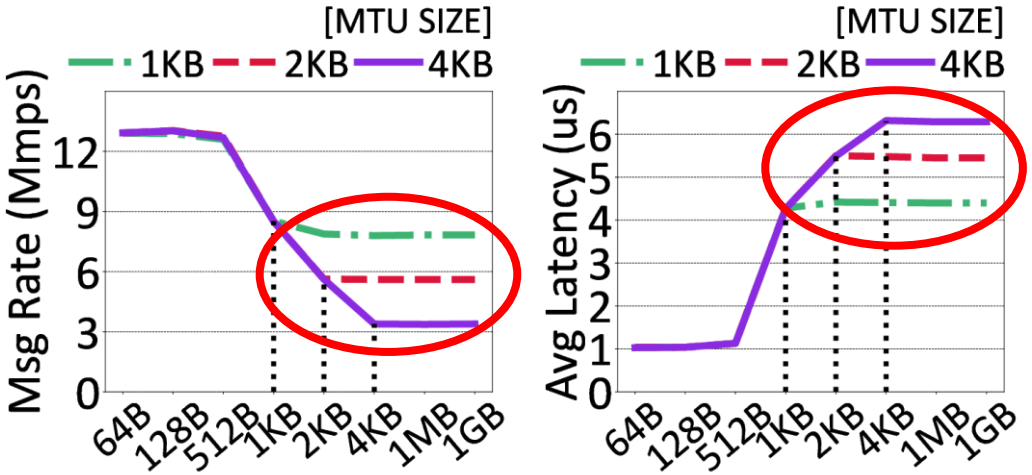
RNIC's Scheduling Mechanism

Setup
- 16-core Intel i7-11700K 3.6GHz
- 32GB RAM
- ConnectX-6 100 Gbps RoCE

Msg-level | QP-level

- 2 apps sharing an RNIC
  - One app sending **small** messages (16B)
  - The other app sending **large** messages as background traffic (64B~1GB)

➔ <u>Performance remains stable once it reaches MTU</u>



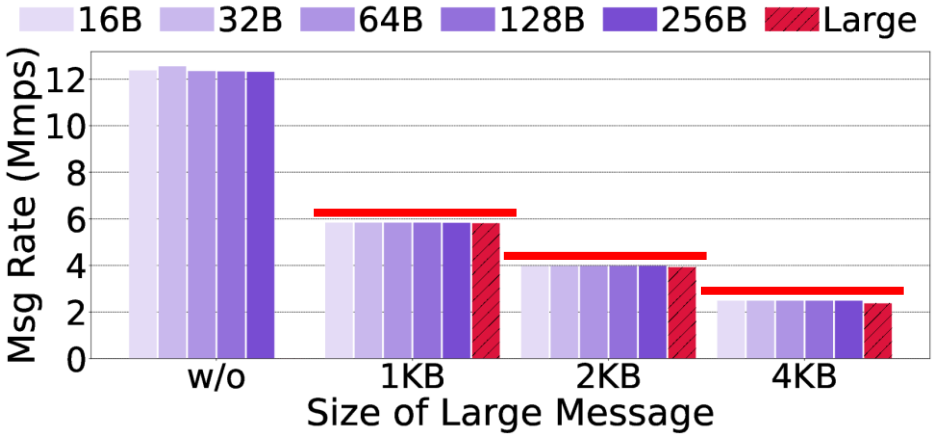
Evenly processing MTU-segmented packets from various QPs

RNIC's Scheduling Mechanism

Setup
- 16-core Intel i7-11700K 3.6GHz
- 32GB RAM
- ConnectX-6 100 Gbps RoCE

Msg-level | QP-level

- 5 apps sending batches of **small messages** (16, 32, 64, 128 and 256B each) with single QP
- Adding an app sending **larger messages** (1, 2, or 4KB) with single QP



➔ Larger messages limits the msg rate of other apps (bandwidth bottleneck)

Round-Robin Scheduling with Multi-QPs

Application Type

Msg-level

	Bandwidth-intensive	Message-intensive	Delay-sensitive
Message Size	Large	Small	Small
Sparsity	Low	High	Low

QP-level

Single QP	B_App_{single}	M_App_{single}	D_App_{single}
Multiple QPs	B_App_{multi}	M_App_{multi}	D_App_{multi}

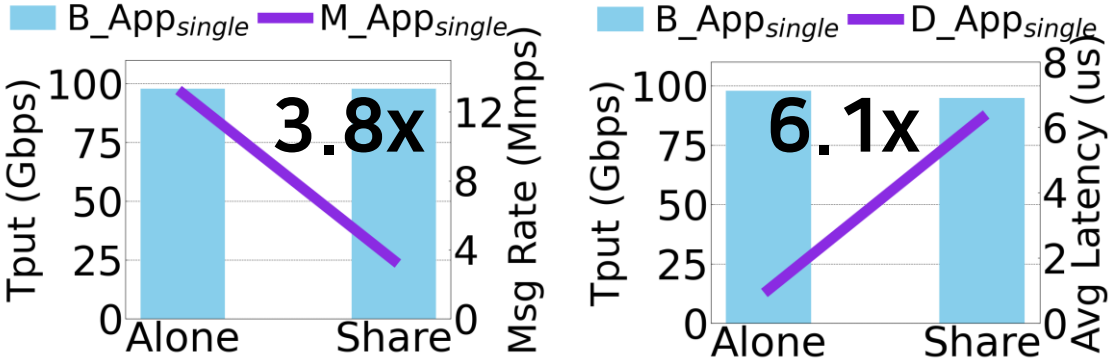
In this paper,

B_App: 1MB

M_App: 16B x 32

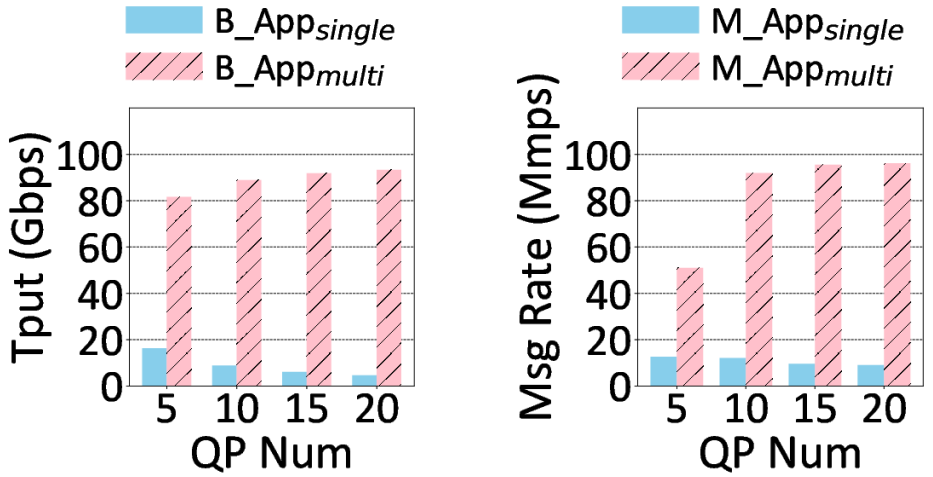
D_App: 16B

Message-level Anomaly



- **B_App** harms both **M_App** and **D_App**.

QP-level Anomaly



- **Multi-QP** app harms **single-QP** app.

RNIC does not support Multi-Tenancy

Prior Works for Performance Isolation

	HW-based	SW-based
Pros	✓ Strict performance isolation	✓ Flexible performance isolation
Cons	- Unable to handle dynamic changes	- Reservation-based resource allocation **(non-work-conserving)** - Unavoidable performance degradation

We propose, **PeRF**

A Preemption-enabled RDMA Framework

- ✓ software-based (flexible)
- ✓ bare-metal performance RDMA (work-conserving)
- ✓ no need for estimation of network resources
- ✓ transparent to applications

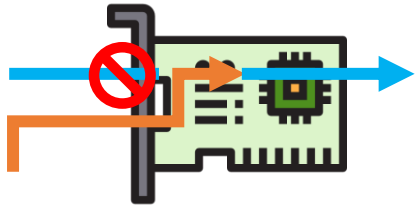
Preemption

preemption : the ability of an operating system to temporarily interrupt a currently running task to run another task.

Preemption in RDMA?

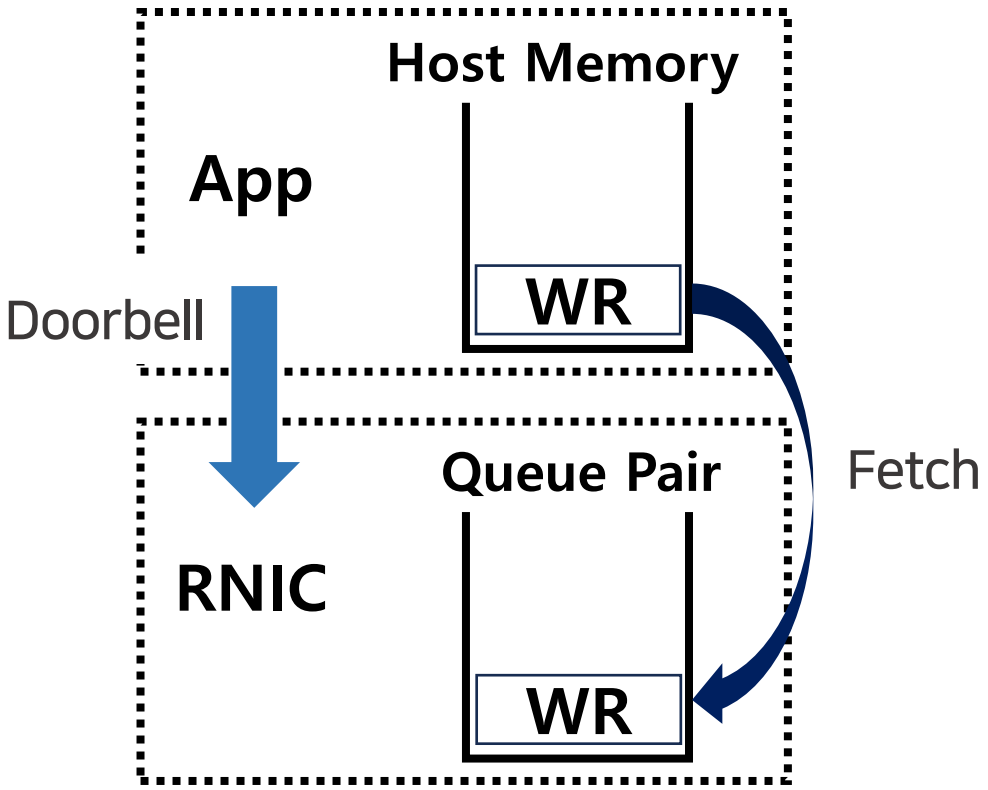
"Interrupting a message and prioritizing another in RNIC"

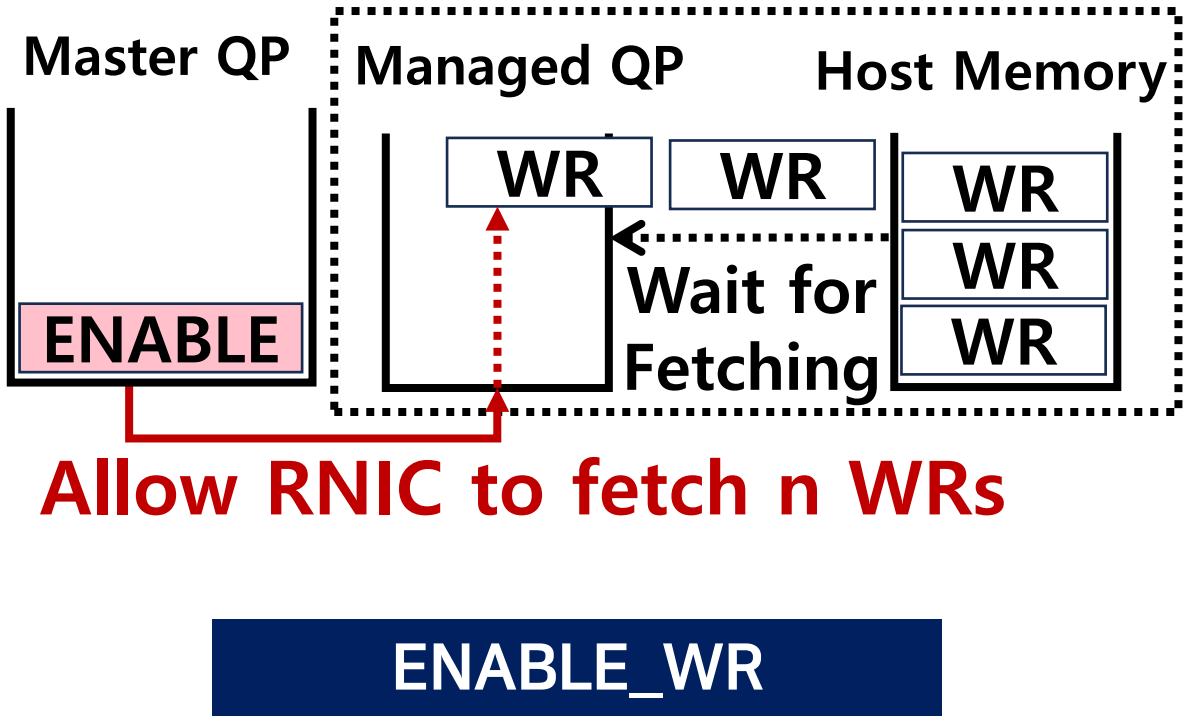
by leveraging a managed QP and a combination of specialized WRs



Preemption Mechanism

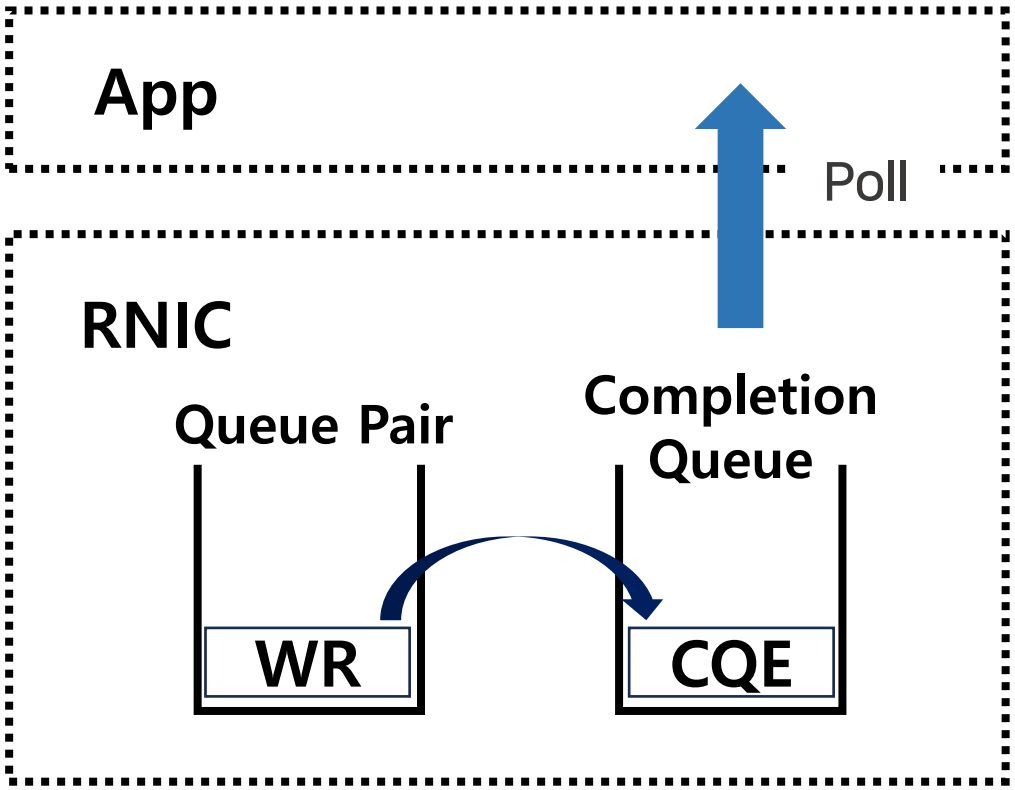
Managed QP and ENABLE_WR

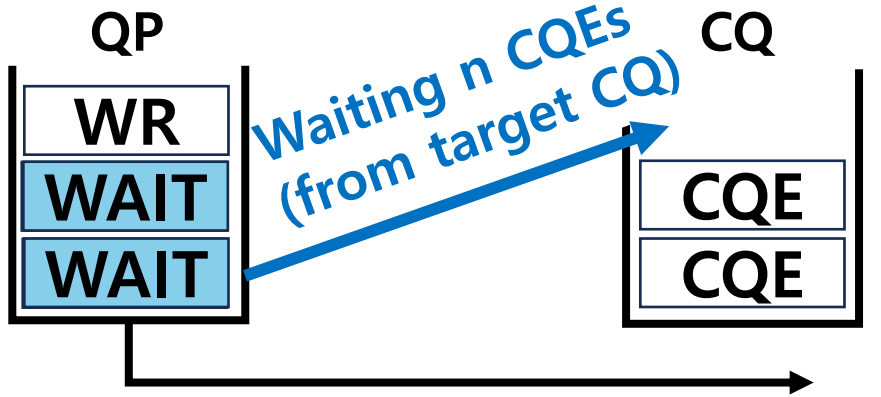




Preemption Mechanism

CQE and WAIT_WR





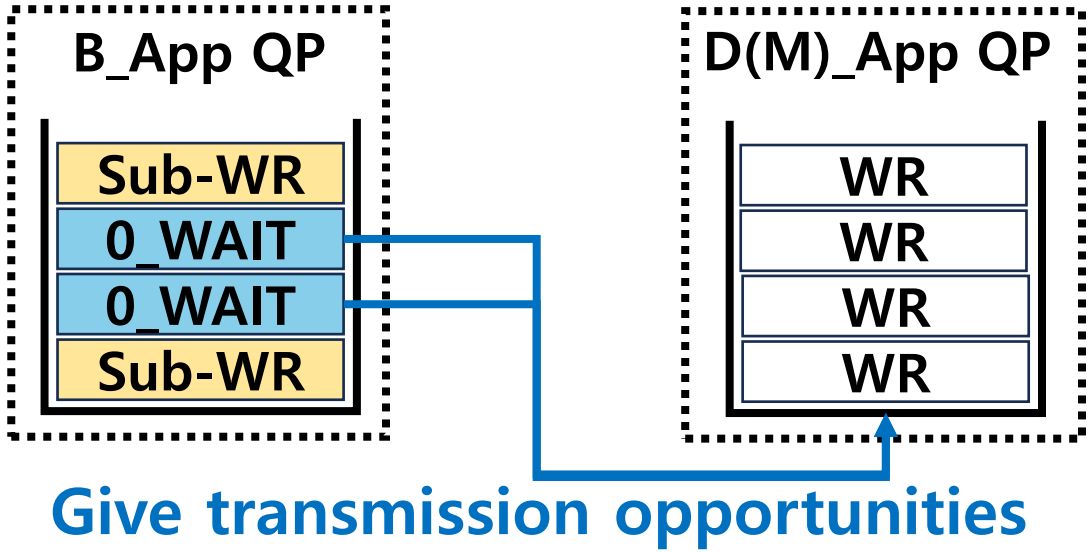
RNIC processes WRs
(until encountering WAIT WR)

WAIT_WR

Preemption Mechanism (Message-level)

Msg-level | QP-level

- **Sub-WR** : subdivided unit of a larger work request
- **0_WAIT** : a WAIT_WR that immediately completes upon execution (wait_cqe_num=0)

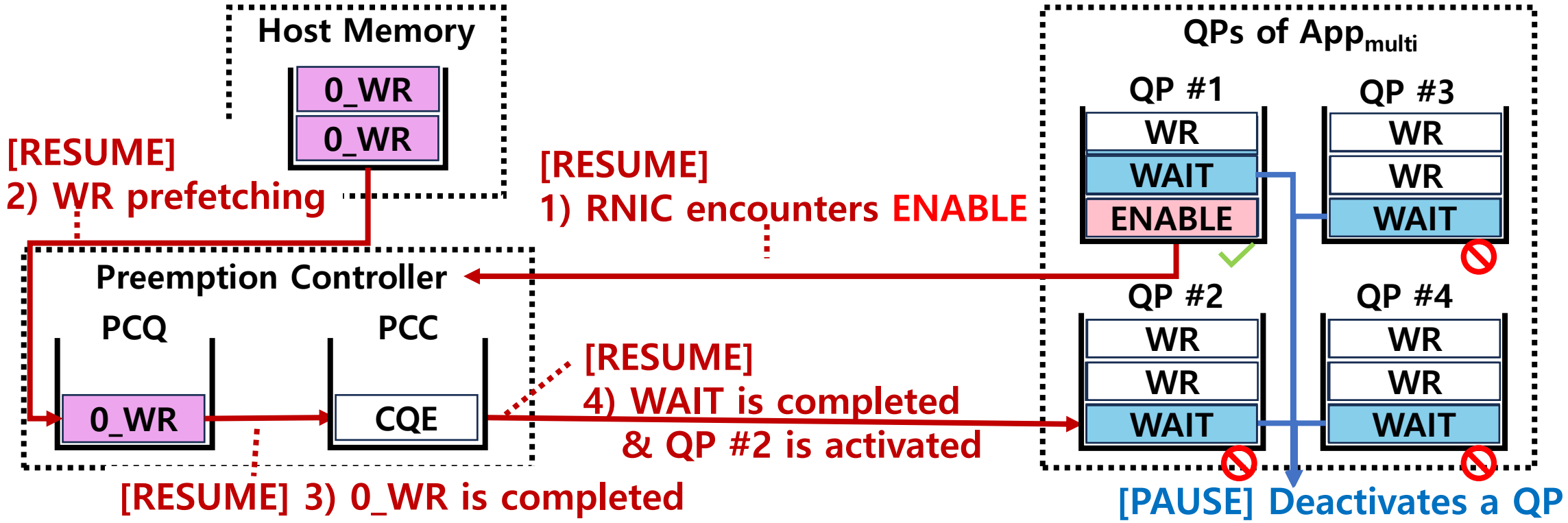


Transmission Interrupt

Preemption Mechanism (QP-level)

Msg-level | QP-level

- 0_WR : a work request with 0 byte message

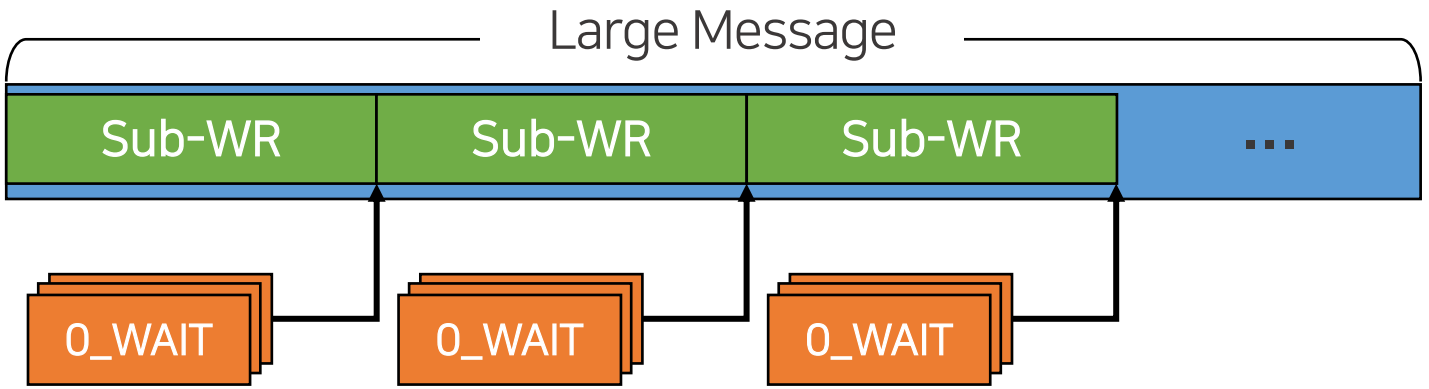


PAUSE/RESUME Operations

Message-level Scheduling

Msg-level | QP-level

Large Message Scheduling Engine (LMSE)



- D(M)_App is competing with B_App
- Split B_App's message into smaller chunks (Sub-WRs)
- Post ***0_WAIT*** WRs between message chunks

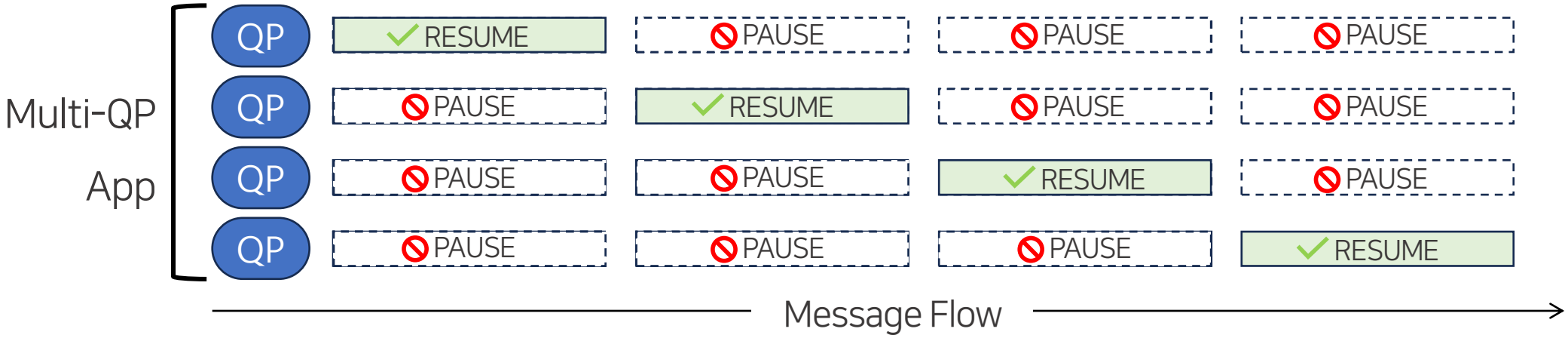
Grant more opportunities for small message flows

QP-level Scheduling

Msg-level | QP-level

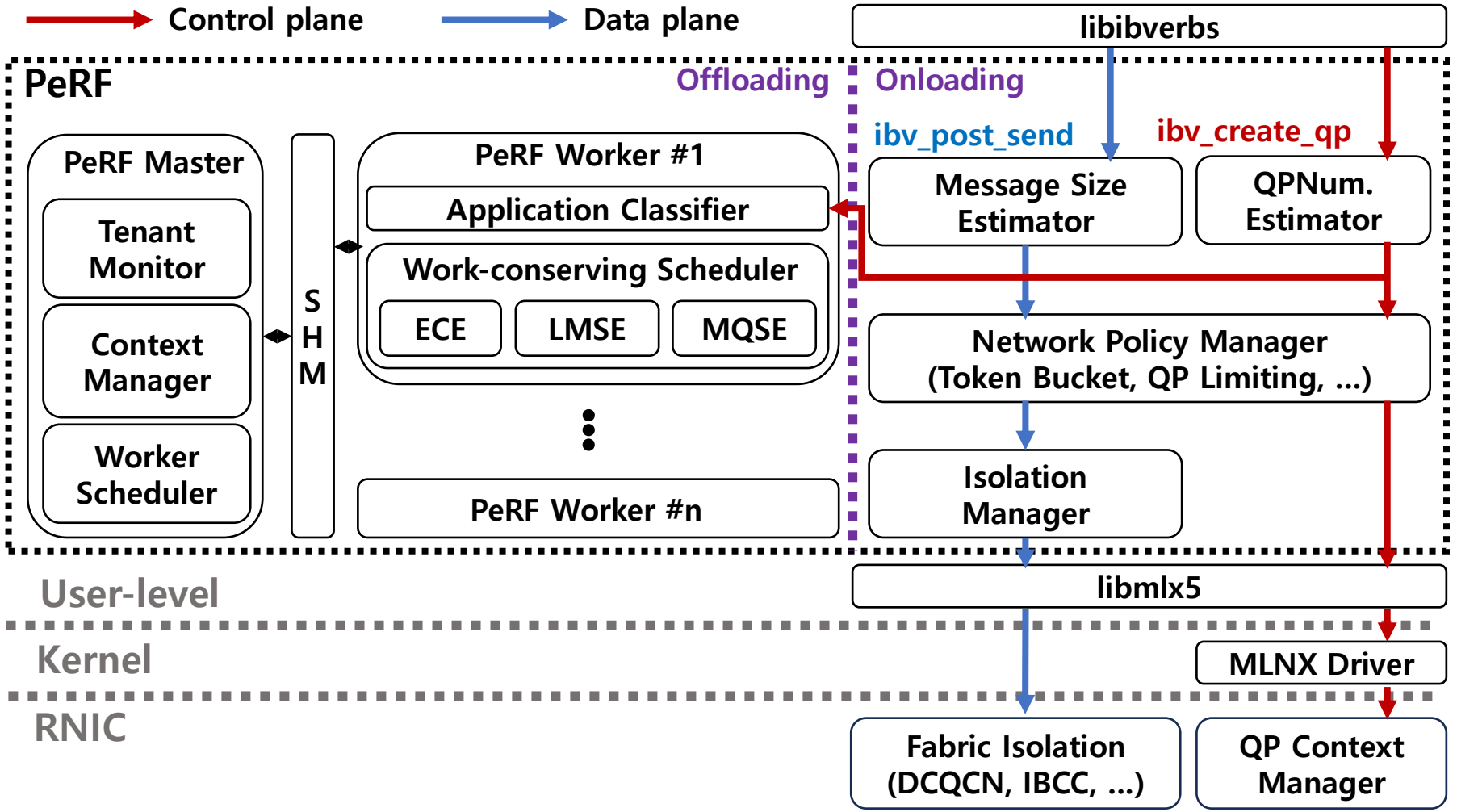
Multi-QP Scheduling Engine (MQSE)

- When competing with a multi-QP App
- Control the number of active QPs with **PAUSE/RESUME** operations



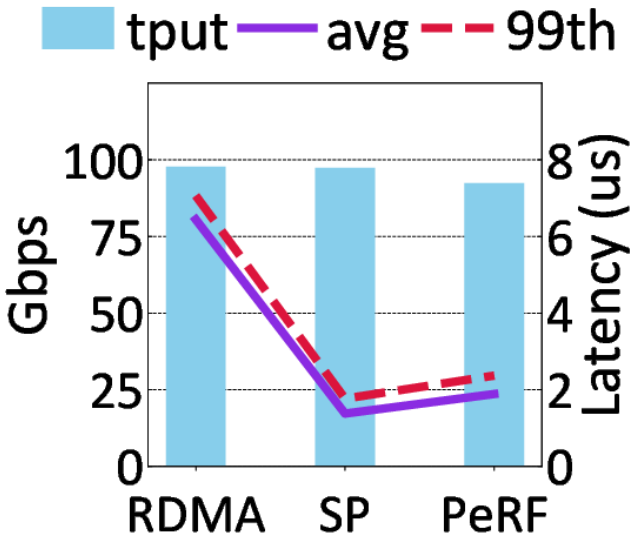
Restrict the number of activated QPs of multi-QP app

PeRF Architecture

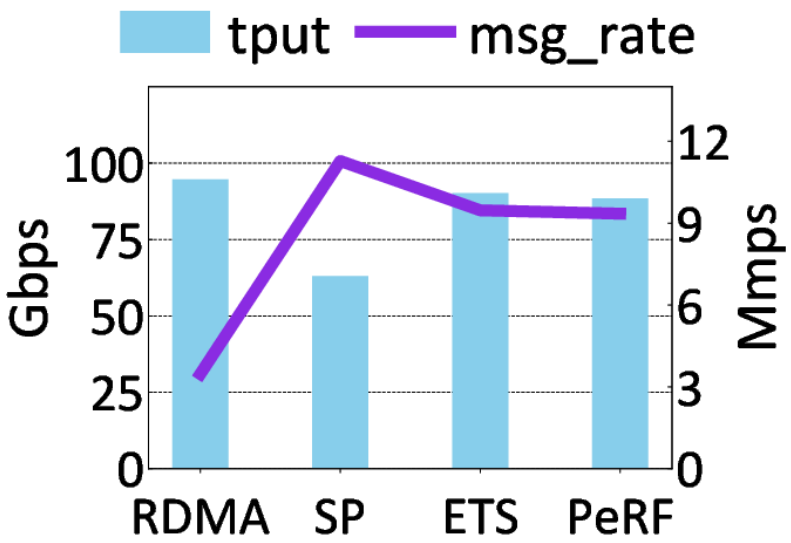


Baseline Benchmark

- **Strict Policy (SP)** : priority queue (set to prioritize D_App)
- **Enhanced Transmission Selection (ETS)** : weighted round-robin (B_App : M_App = 16 : 1)



B_App_{single} vs D_App_{single}



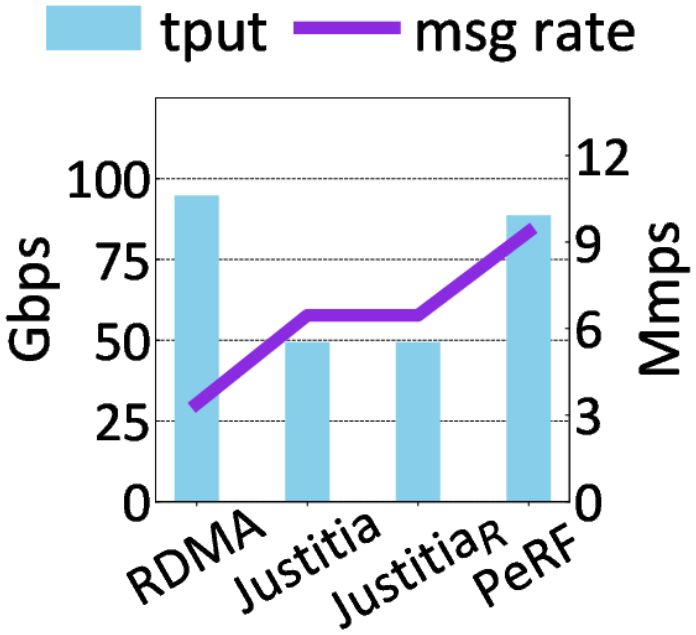
B_App_{single} vs M_App_{single}

PeRF performs nearly as HW-based solutions

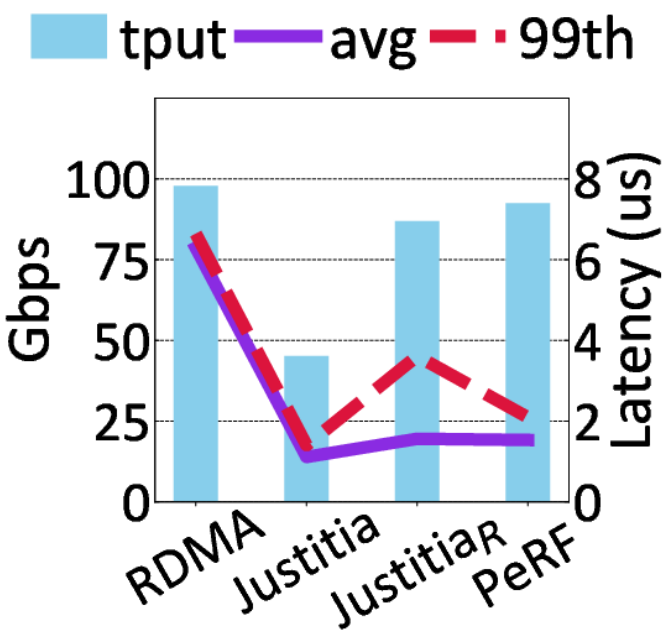
Message-level Isolation

Msg-level | QP-level

- **Justitia** : a token-based resource allocation solution
- **Justitia$_R$** : a relaxed version of Justitia (latency threshold = 15 us)



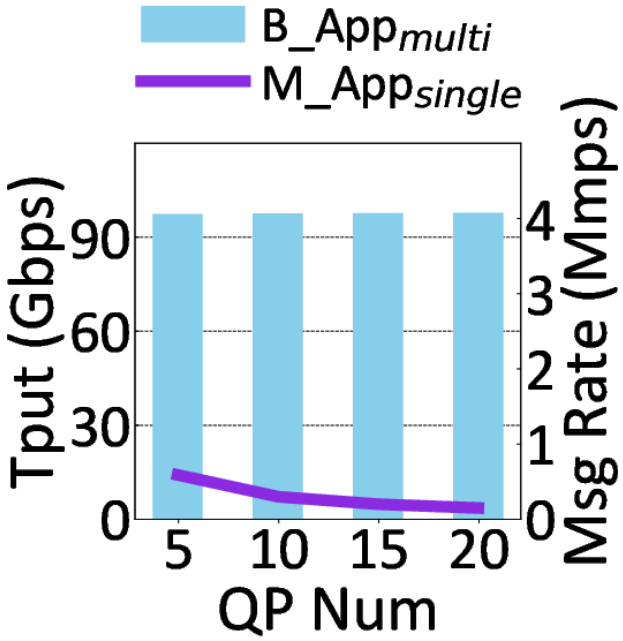
B_App$_{single}$ vs M_App$_{single}$



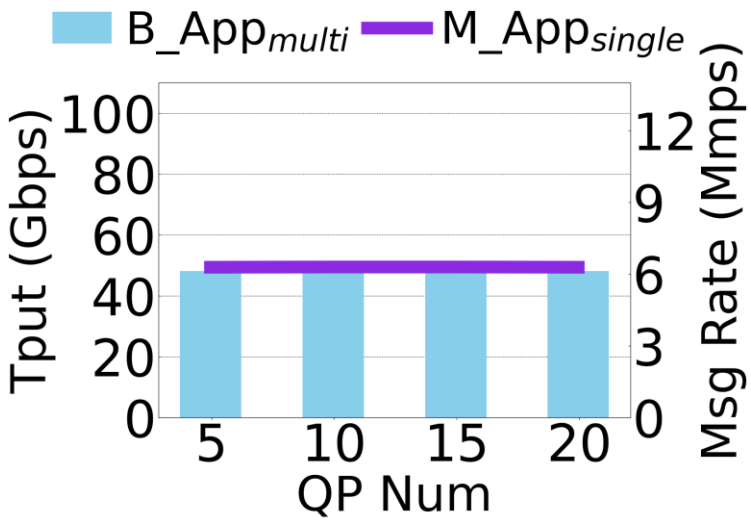
B_App$_{single}$ vs D_App$_{single}$

QP-level Isolation

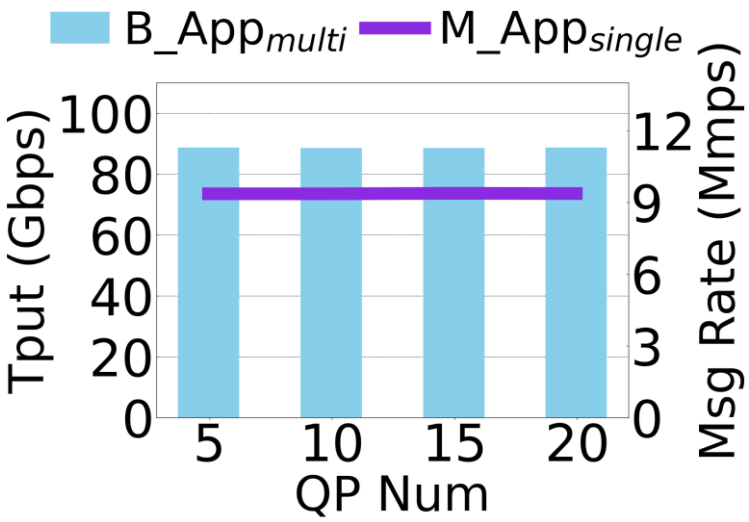
Msg-level | QP-level



RDMA



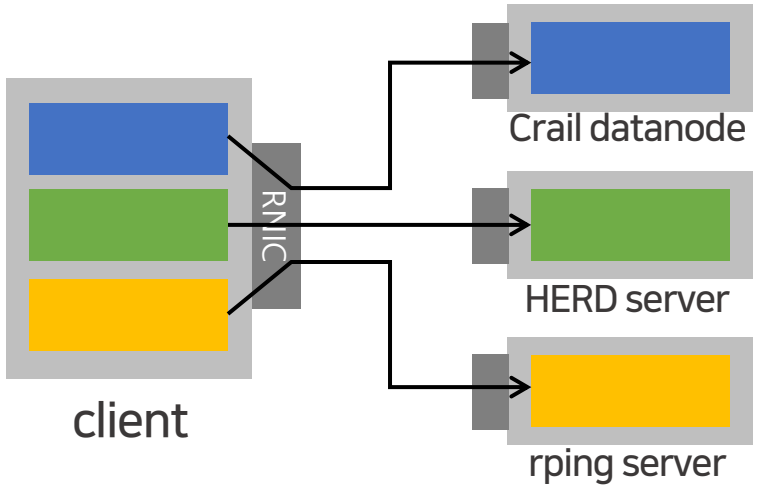
Justitia

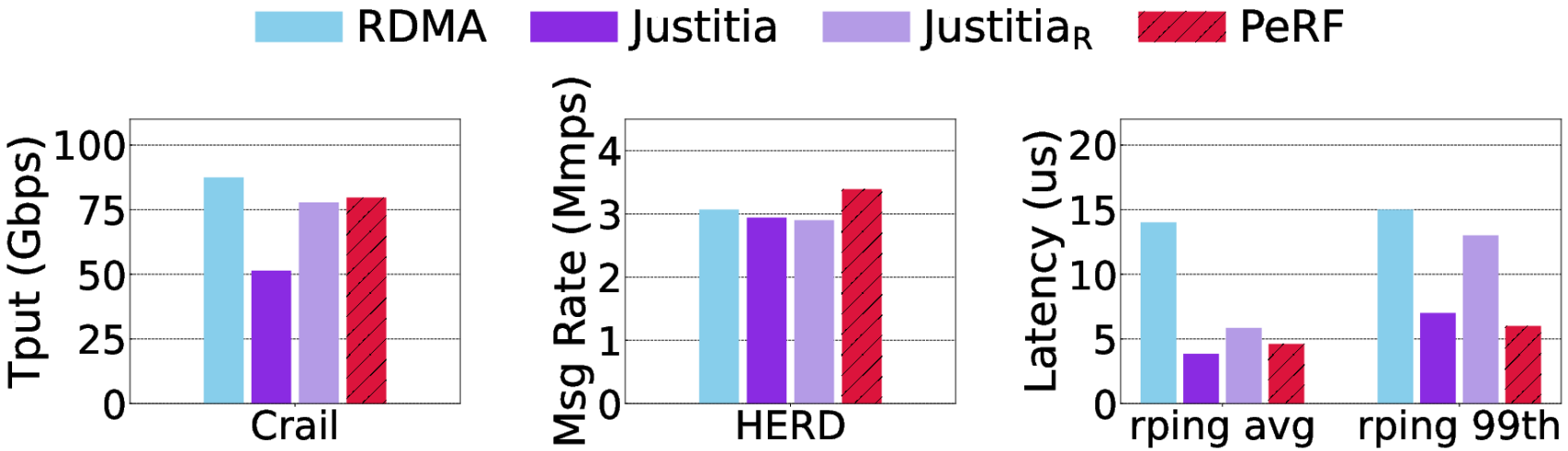


PeRF

Real World Applications

Application	Type	Message Size	Batch
Apache Crail	B_App	1GB (1MB chunk)	1
HERD	M_App	5% PUT 50B 95% GET 17B	32
rping	D_App	16B	1





Other Experiments

1. Other Commercial RNICs Test
2. Support for SEND and READ Operations
3. Scalability Test
4. Weighted Policy Test
5. Congested Networks Test

... and more!

PeRF

- ✓ Traditional RDMA solutions struggle with **performance anomalies** and **inefficient resource utilization** in **multi-tenant** environments.
- ✓ PeRF uses a novel RNIC **preemption mechanism** and **work-conserving packet/QP scheduling** to address these challenges.
- ✓ PeRF ensures **software-based performance isolation** without sacrificing the **high performance** of RDMA.

Thanks & Q/A

github: https://github.com/acryl-aaai/perf
contact: ① sglee0323@gmail.com
② mingyuchoi514@gmail.com