IEEE CloudNet 2020
November 9, 2020

# Roadblocks of I/O Parallelization: Removing H/W Contentions by Static Role Assignment in VNFs

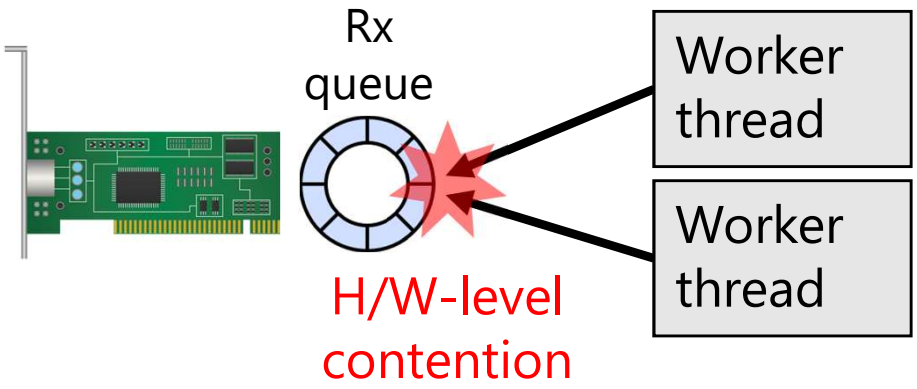
**Masahiro Asada**[†1] Ryota Kawashima[†1] Hiroki Nakayama[†2]
Tsunemasa Hayashi[†2] Hiroshi Matsuo[†1]

[†1] Nagoya Institute of Technology
[†2] BOSCO Technologies Inc.

# Summary



- Problem: Roadblocks of packet I/O parallelization
  - Sharing a single Rx queue




Trivial arbitration cost

Increased CPU cycles due to unexplicit H/W-level contention

- Proposal: Static role assignment
  - Based on careful analyses of packet receiving mechanism
  - Independent from specific H/W features

Contribution

Flatten CPU cycle consumptions
→ Minimize S/W-side overhead

Remaining Problem

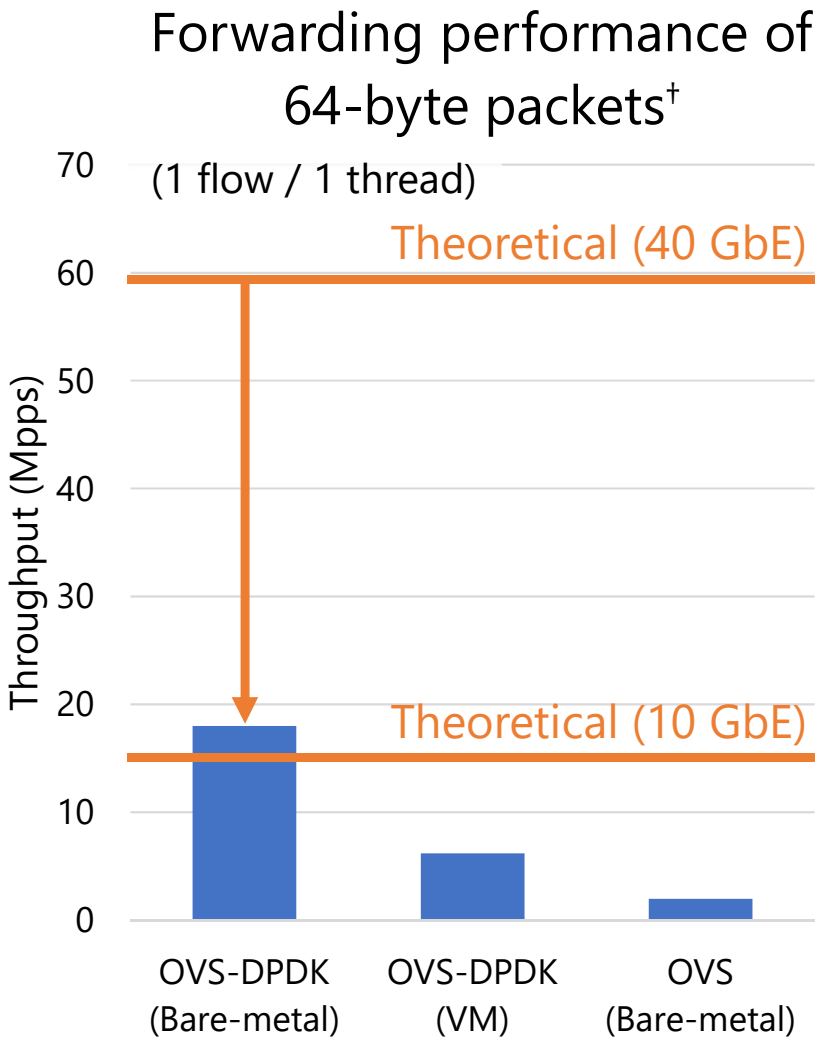
Throughput is not improved
due to H/W-side limitation

# Contents

# Requirement: Boost Performance



- >10 Gbps is challenging to reach




DPDK: Data Plane Development Kit
OVS: Open vSwitch

- >800 Gbps Ethernet comes
- Limited per-core performance

100 Gbps with 64-byte packets

Processing Time: 5.12 nsec/packet
CPU cycles: ≦17 cycle/packet ††

**I/O Parallelization is needed**

† Evaluation of Forwarding Efficiency in NFV-nodes toward Predictable Service Chain Performance.
R. Kawashima et al.
IEEE Transactions on Network and Service Management, 2017

†† Make the most out of last level cache in intel processors.
A. Farshin et al.
Proceedings of the Fourteenth EuroSys Conference, 2019

# Requirement: Keep Flexibility



- Hardware accelerators are not silver bullets
  - e.g. SmartNIC with FPGA

Pros.

- Optimize specific workloads

Cons.

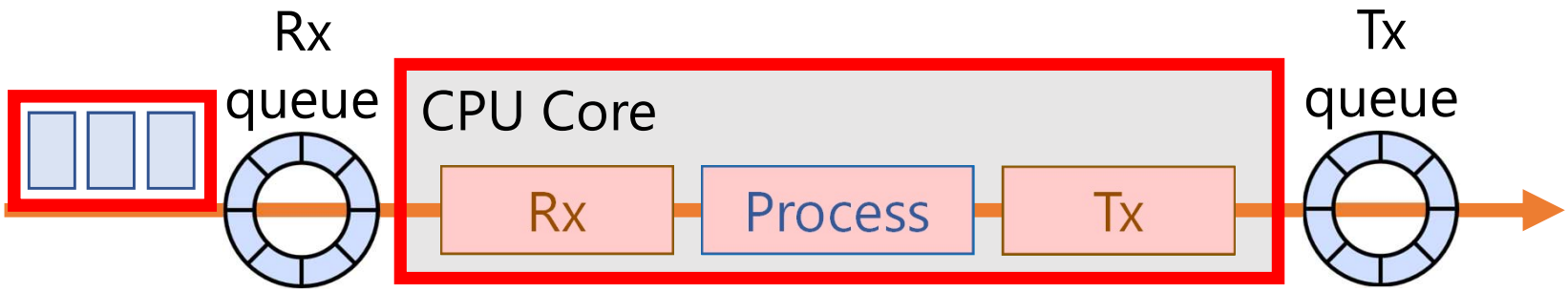
- Inflexible deployment
  - Long time to deploy
  - Difficult to share H/W resources
- Costly development
  - Vendor-specific tools, procedures

Network flexibility should be kept by software

# Contents

# Packet Processing with DPDK






```
processing_loop()
{
  N = 32;  /* Default */
  packets[N];
  rx_queue;
  tx_queue;

  while (true)
  {
    rx_burst(rx_queue, packets);
    process(packets);
    tx_burst(tx_queue, packets);
  }
}
```

Packet processing loop

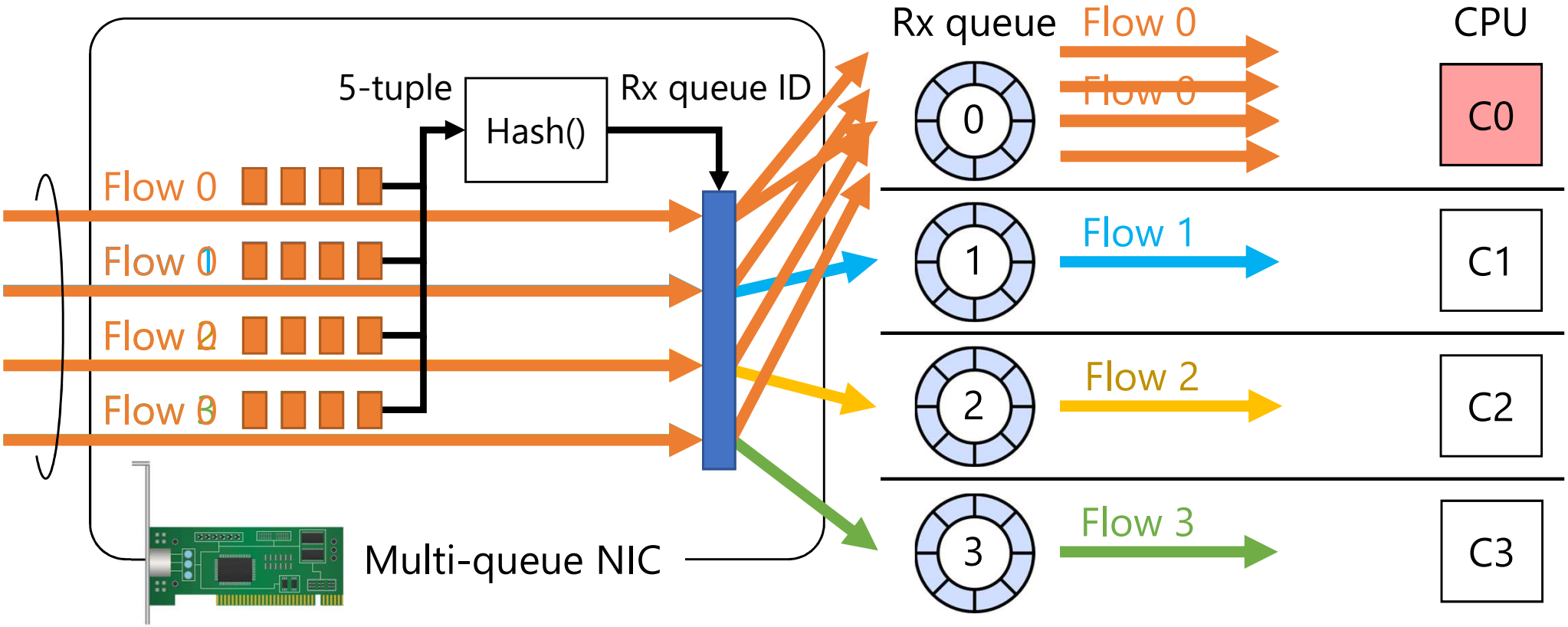
- Concept
  - 1 core ⇔ 1 thread
  - Packet batching
- For performance improvement
  - Batch many packets
  - Reduce CPU cycles

# Flow-level Parallelization



- RSS: Receive Side Scaling
  - Distribute packets to cores with the hash value of 5-tuple†

† Src IP / Dst IP / Src port / Dst port / Protocol number



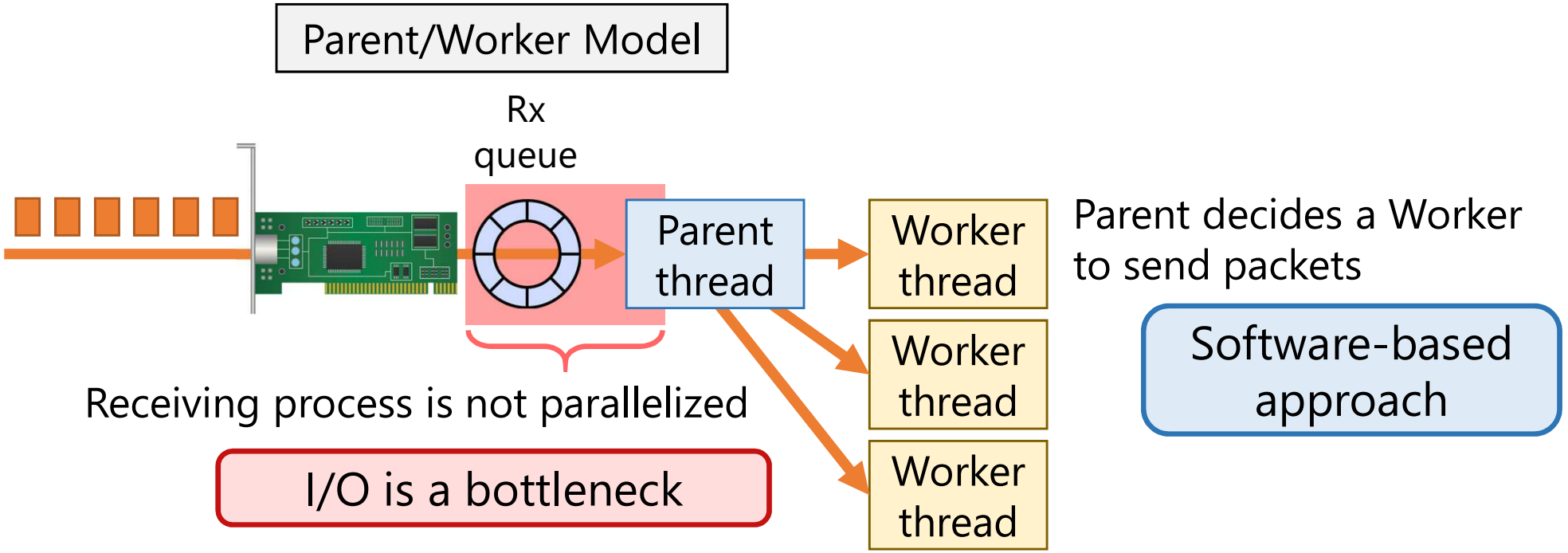

- Per-flow performance is not improved
  - Not work with a single-flow traffic

**Finer-grained parallelization than flows is needed**

# Packet-level Parallelization



- Multiple packet processing models†



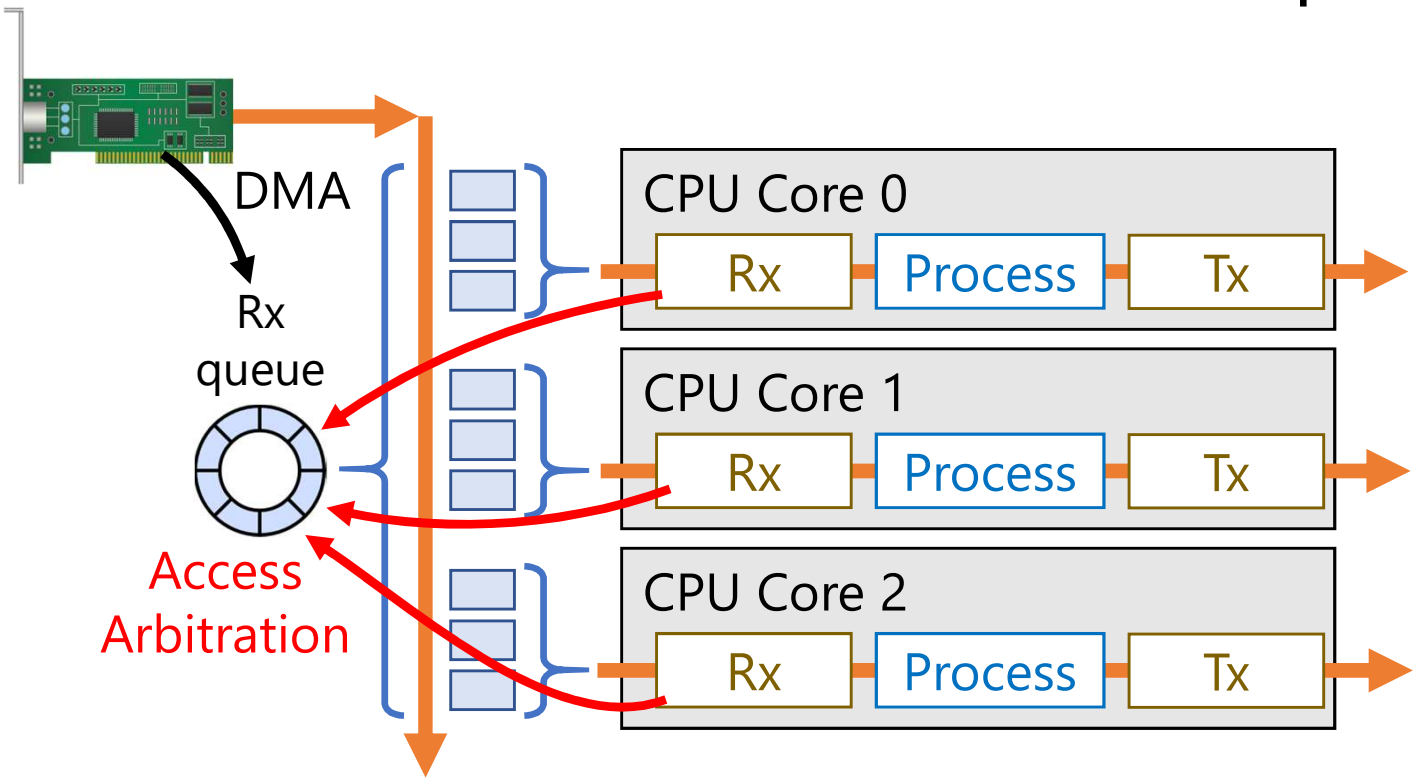

I/O must be included in parallelization

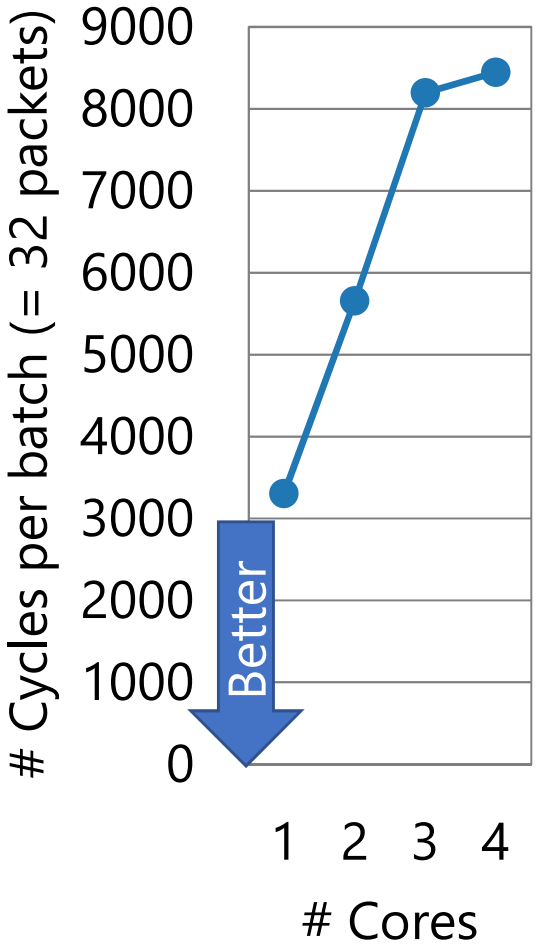
† Designing Virtual Network Functions for 100 GbE Network Using Multicore Processors.
P. Li et al.
2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)

# Simple I/O Parallelization



- Added arbitration to a receive queue†




† Datapath Parallelization for Improving the I/O Performance on NFV Nodes (in Japanese).
M. Asada, R. Kawashima, H. Nakayama, T. Hayashi, and H. Matsuo
IEICE Technical Report (NS2019-37), 2019

Unexpected growth of overheads

**Considering H/W behaviors is needed for I/O parallelization**

# Contents



- Background
- Parallelization Schemes
- **Proposal**
- Evaluation
- Discussion, Conclusion

# Requirements

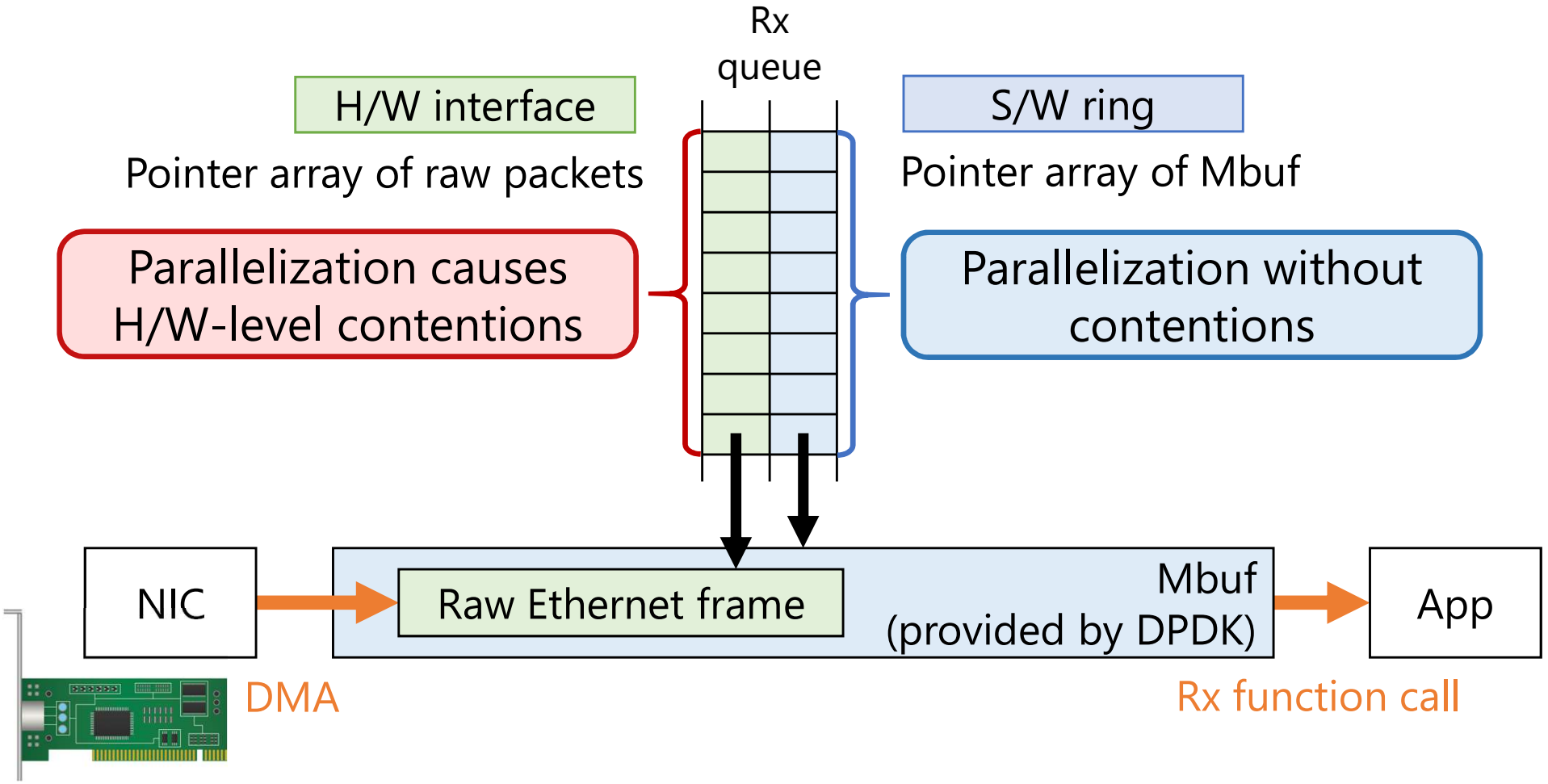


- Fast Datapath by I/O Parallelization
  - Utilize performance of multi-core processors
  - Utilize DPDK's packet batching
- Independent from Specific Hardware Features
  - Enable familiar resources for flexibility
    - Development tools
    - Programming languages
    - OS, API
    - Servers
- Rx-Mechanism Awareness
  - Consider the interaction with H/W

# H/W Contentions in A Receive Queue

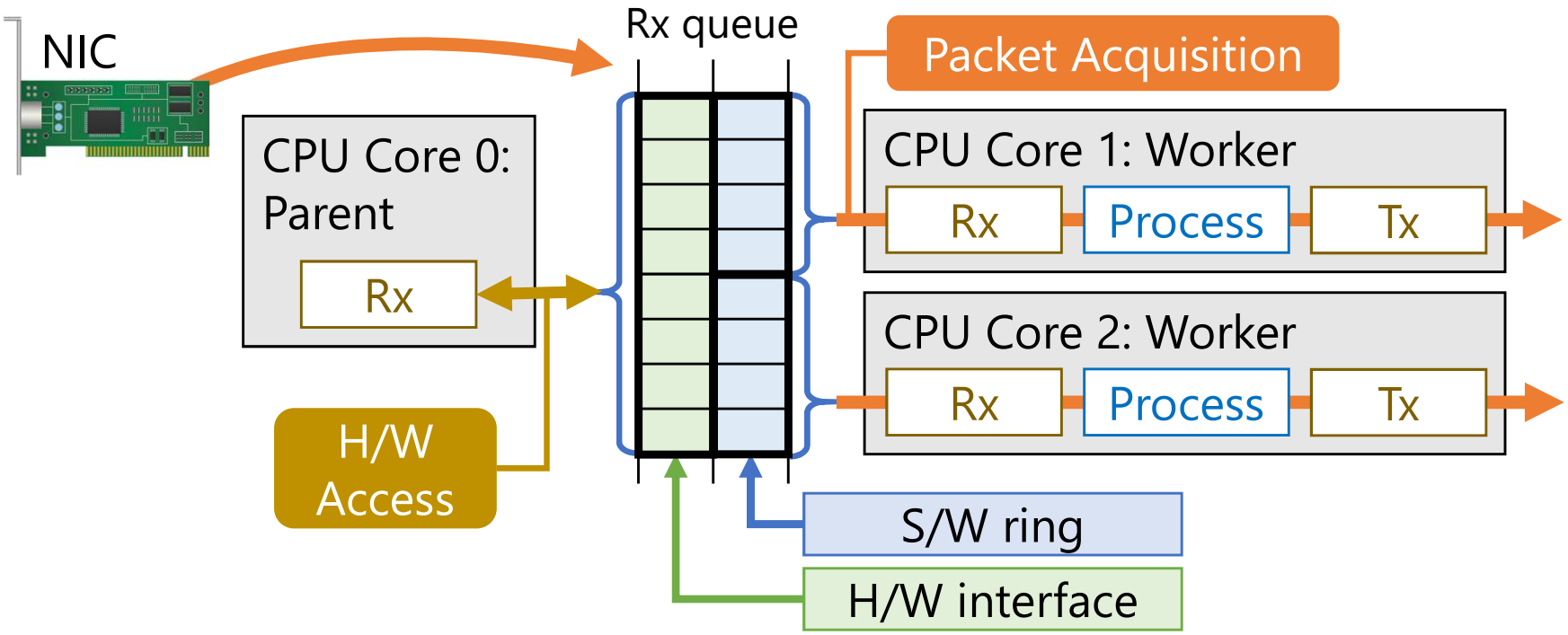


- Two aspects in a receive queue
  - H/W interface: Interactions with a NIC
  - S/W ring: Software-friendly data structure of packets



# Proposal: Static Role Assignment

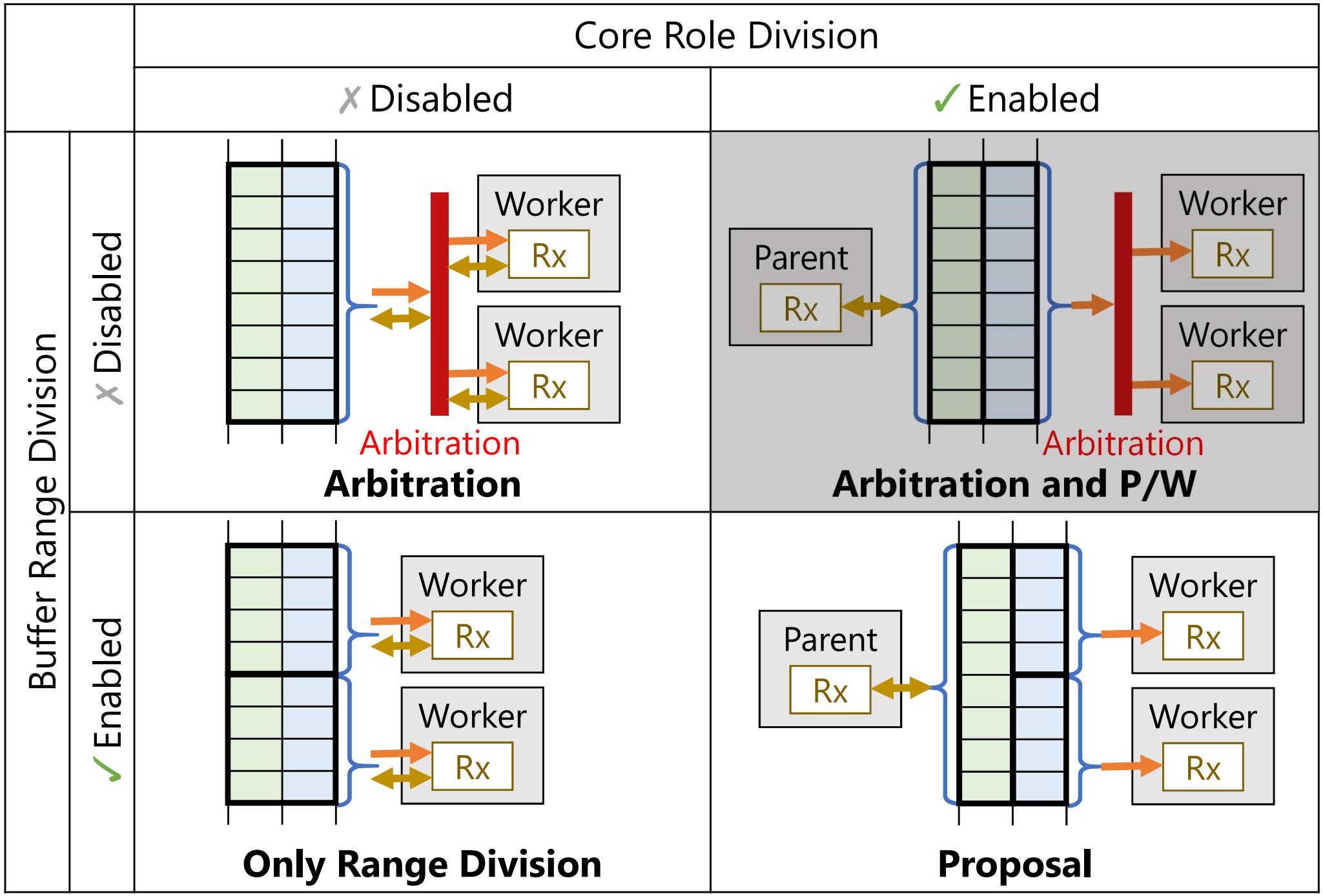





- Role of CPU cores (threads)
  - Parent: H/W Access — Executed by a single thread to avoid H/W contentions
  - Worker: Packet Acquisition — Pure S/W operations which can be parallelized
- Assumed applications
  - Stateless per-packet processing (e.g. routing, address translation, encapsulation)

# Comparation of Parallelization Patterns





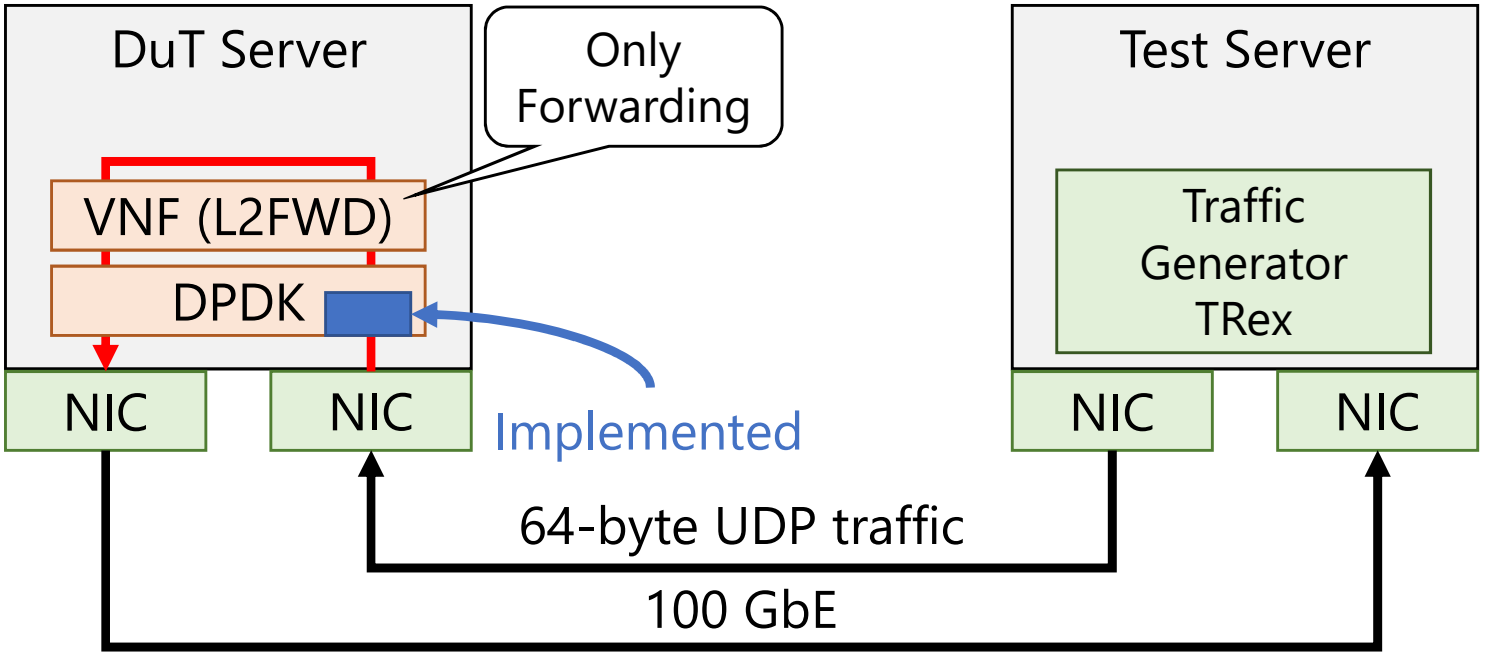
# Contents



- Background
- Parallelization Schemes
- Proposal
- **Evaluation**
- Discussion, Conclusion

# Evaluation Environment

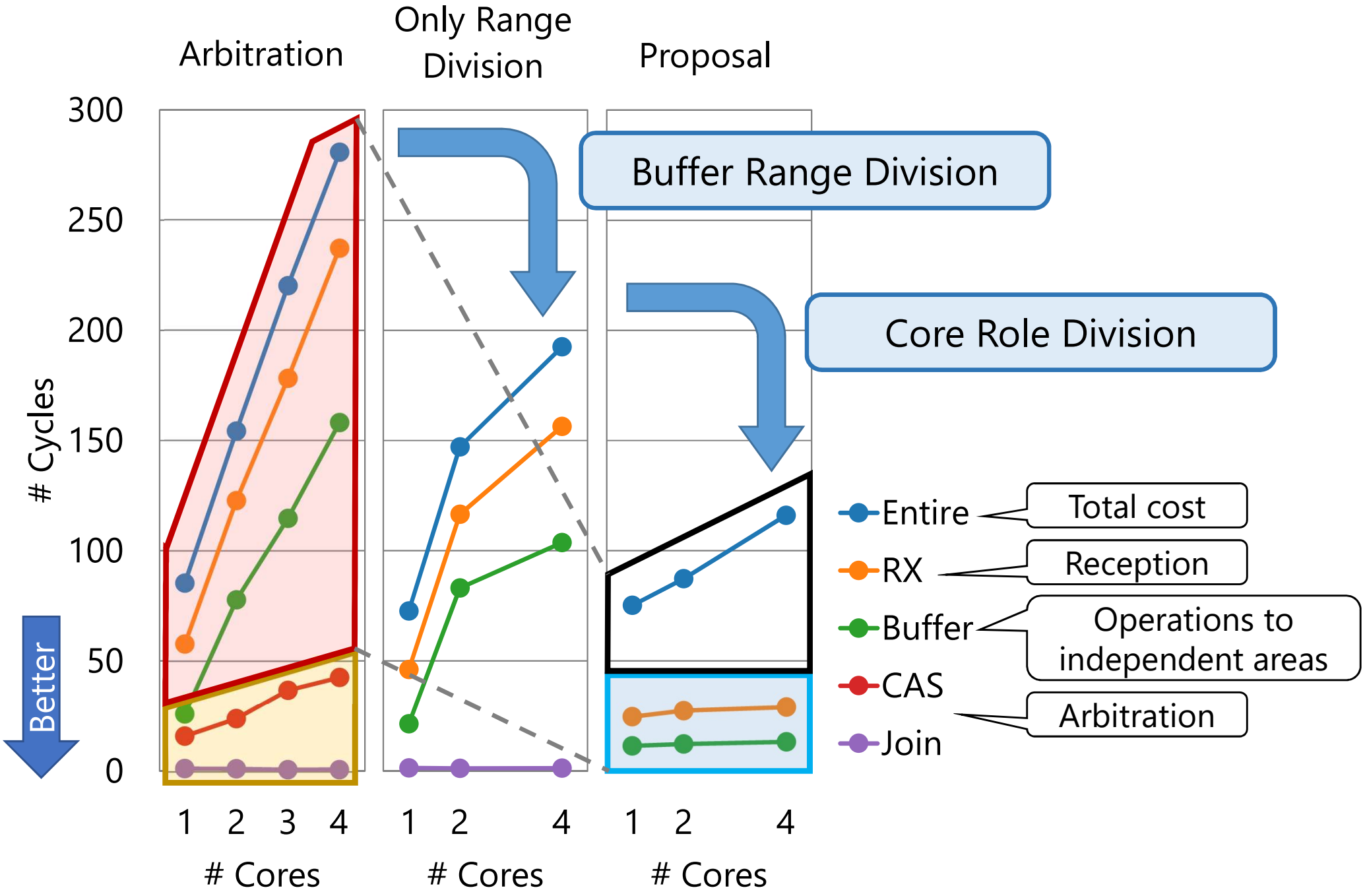





| | DuT Server | Test Server |
|---|---|---|
| CPU | Intel Core i9-7940X @3.10GHz<br>14 cores (HT disabled) | Intel Core i7-7900X @3.30GHz<br>10 cores (HT disabled) |
| Mem | 32 GB DDR4 | 64 GB DDR4 |
| NIC | Mellanox Technologies ConnectX-5 Ex 100 GbE Dual-Port | |
| OS | CentOS 7.7 | CentOS 7.7 |
| DPDK | v19.11 | v19.05 |
| TRex | — | v2.56 |

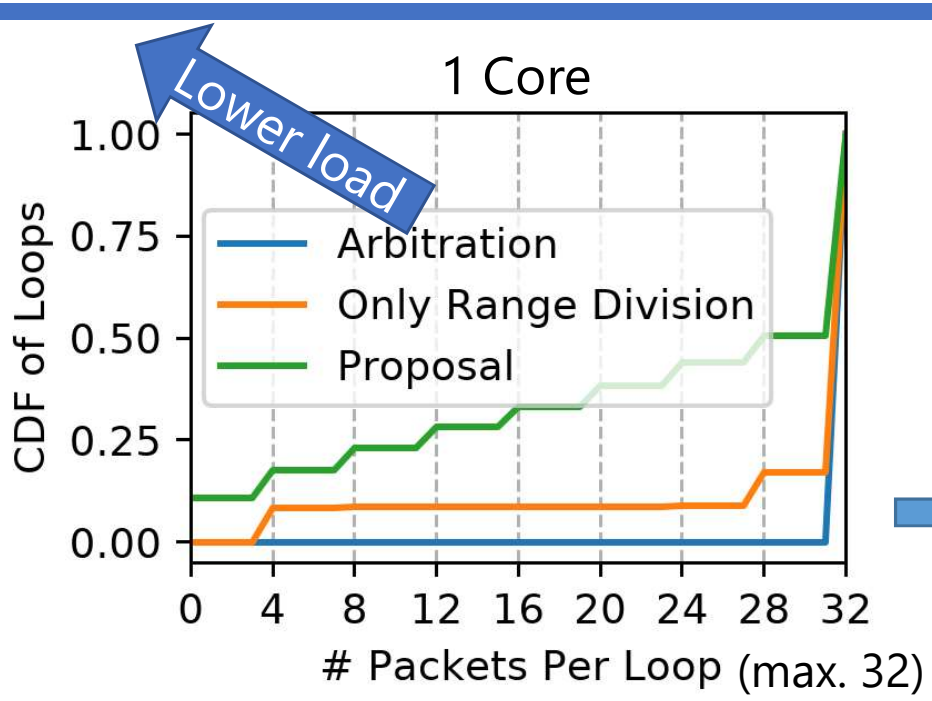
# Consumed CPU Cycles per Packet

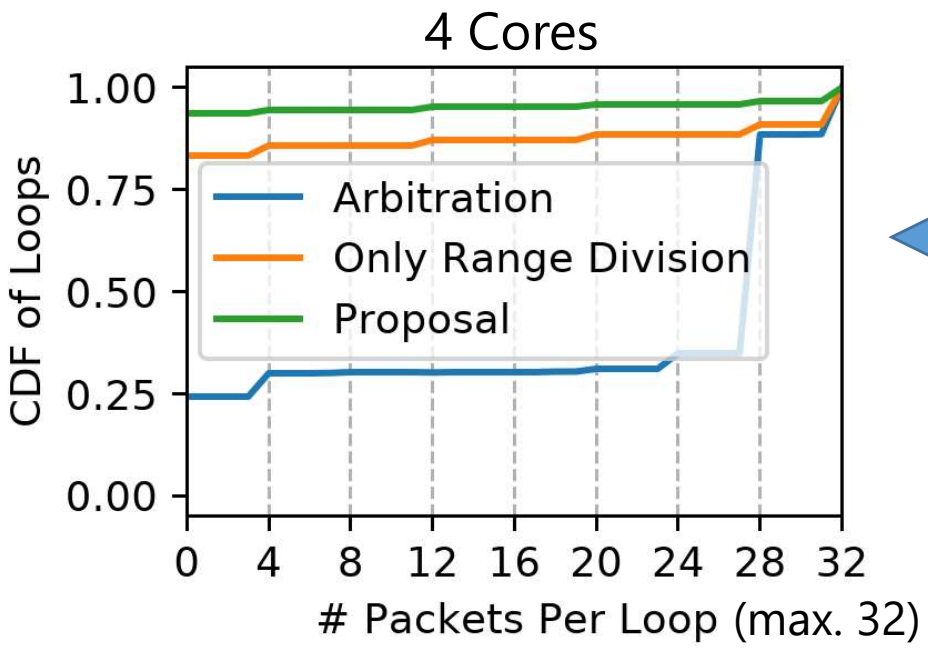


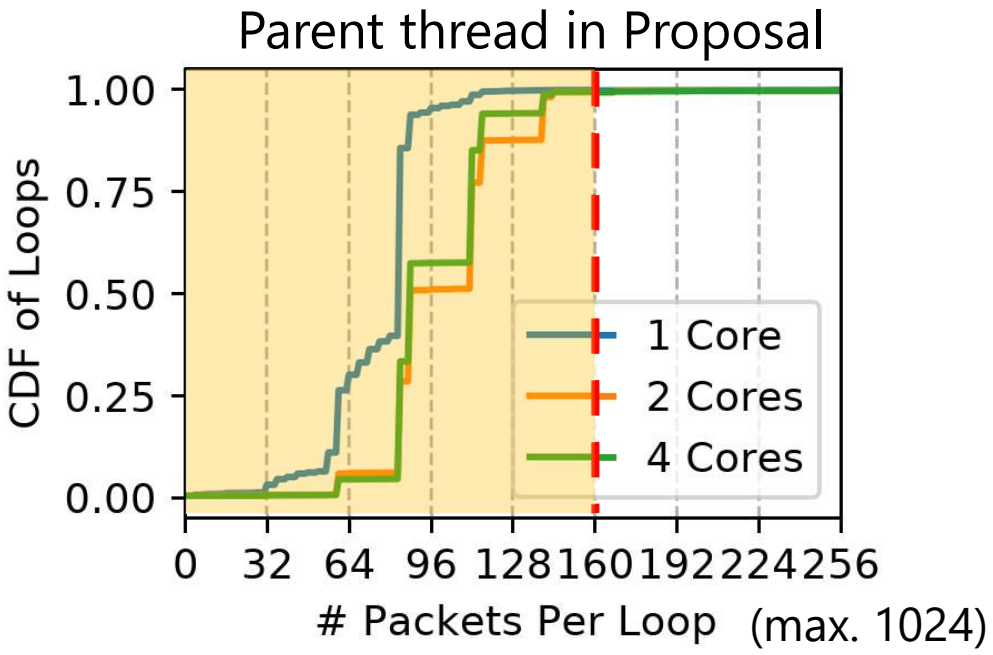


# Number of Processed Packets





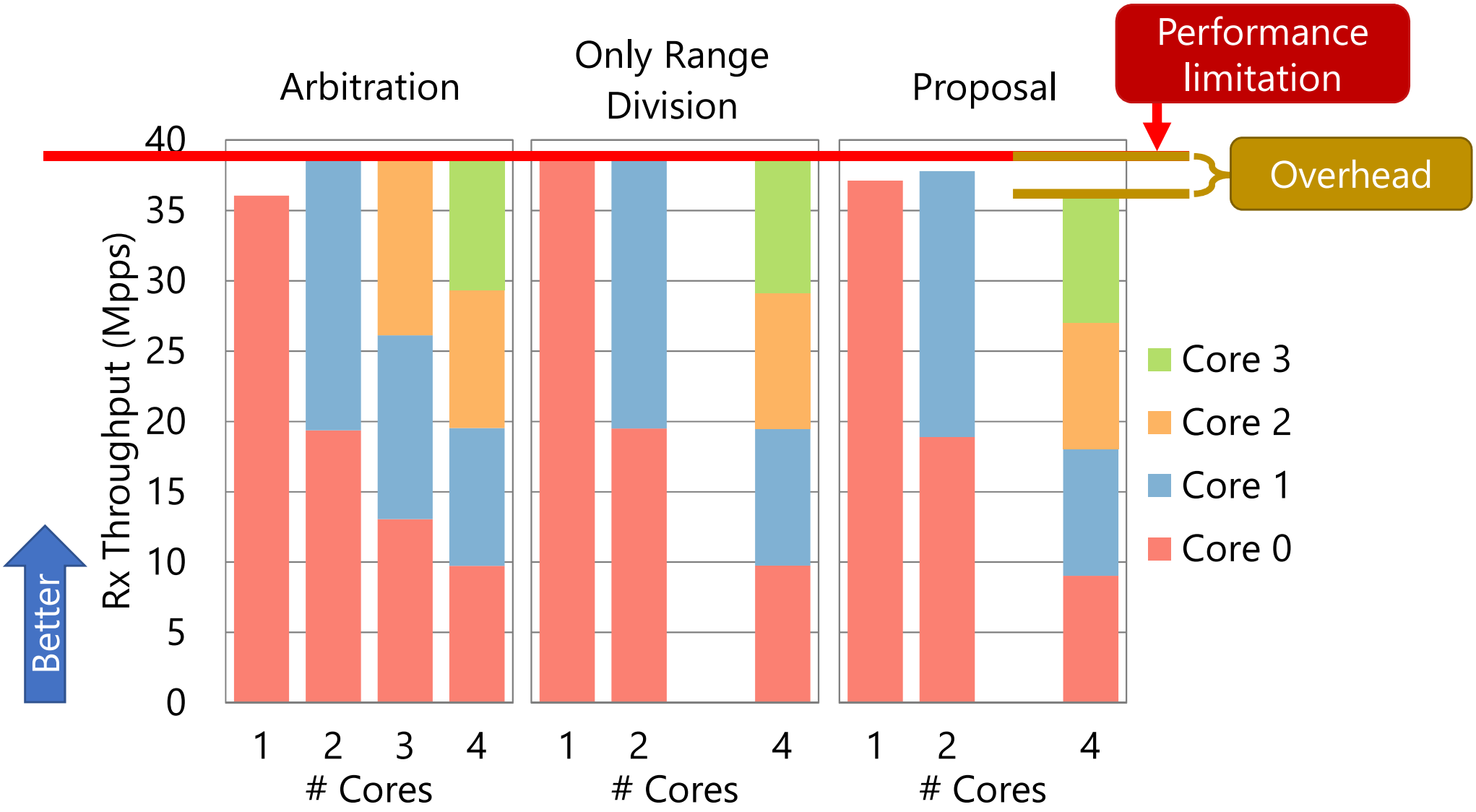

Lower Load

S/W is not a bottleneck

CDF: Cumulative distribution function

# Throughput






S/W is not a bottleneck

H/W-side limitation

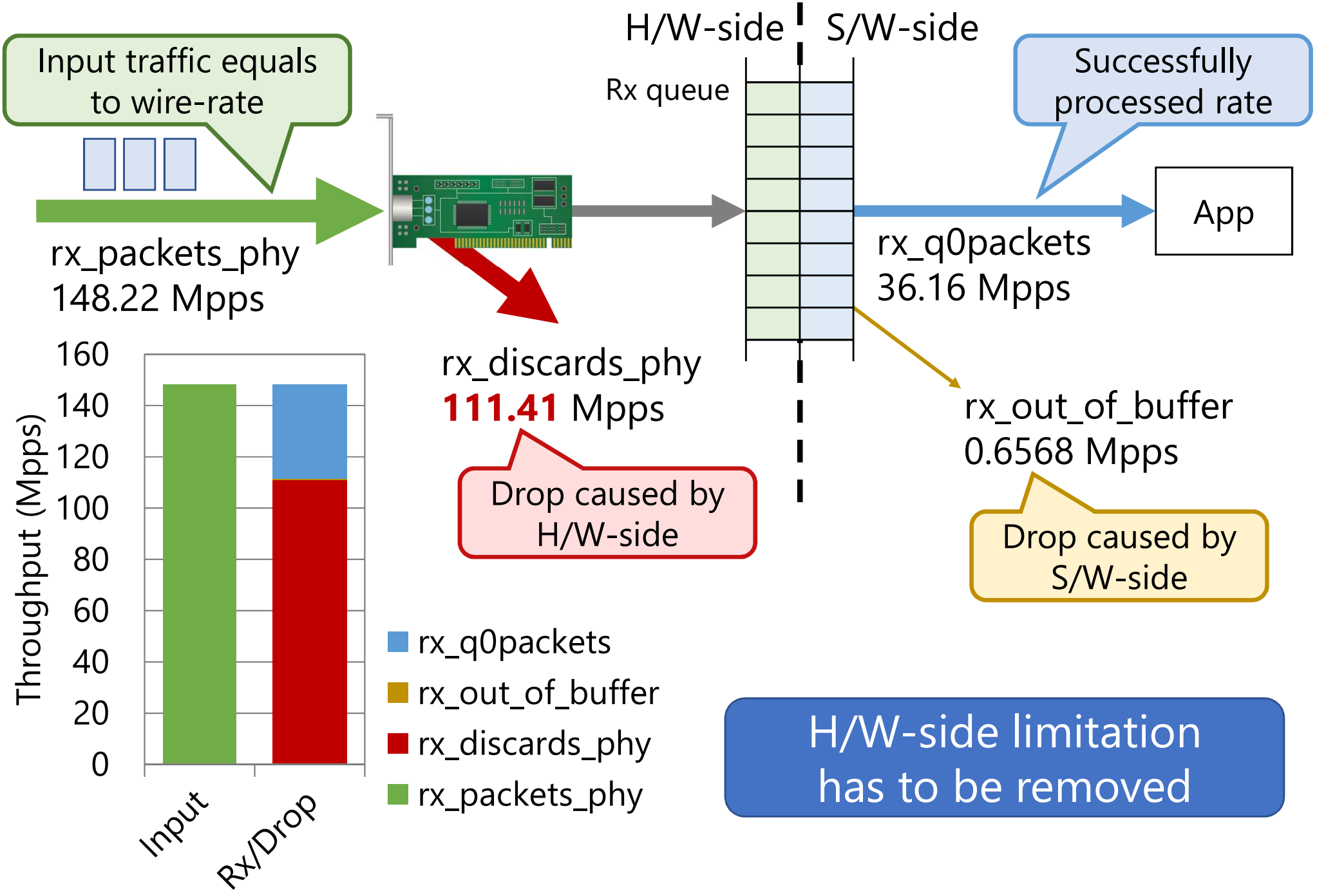
# Contents



- Background
- Parallelization Schemes
- Proposal
- Evaluation
- **Discussion, Conclusion**

# Packet Drop inside H/W





# Conclusion



- Roadblocks of I/O parallelization in VNFs
  - Unexplicit H/W-level contentions
- Proposal: Static role assignment
  - Based on the analysis of packet reception mechanism
  - Removed most of overheads in S/W-side
  - H/W-side limitation obstructs the improvement of throughput
- Future work
  - Further investigation and optimization for linear scaling