PRISM: Rethinking the RDMA Interface for Distributed Systems

Matthew Burke, Sowmya Dharaniparagada, Shannon Joyner, Adriana Szekeres, Jacob Nelson, Irene Zhang, Dan R. K. Ports

SOSP'21

2022. 04. 12

Presentation by Han, Yejin

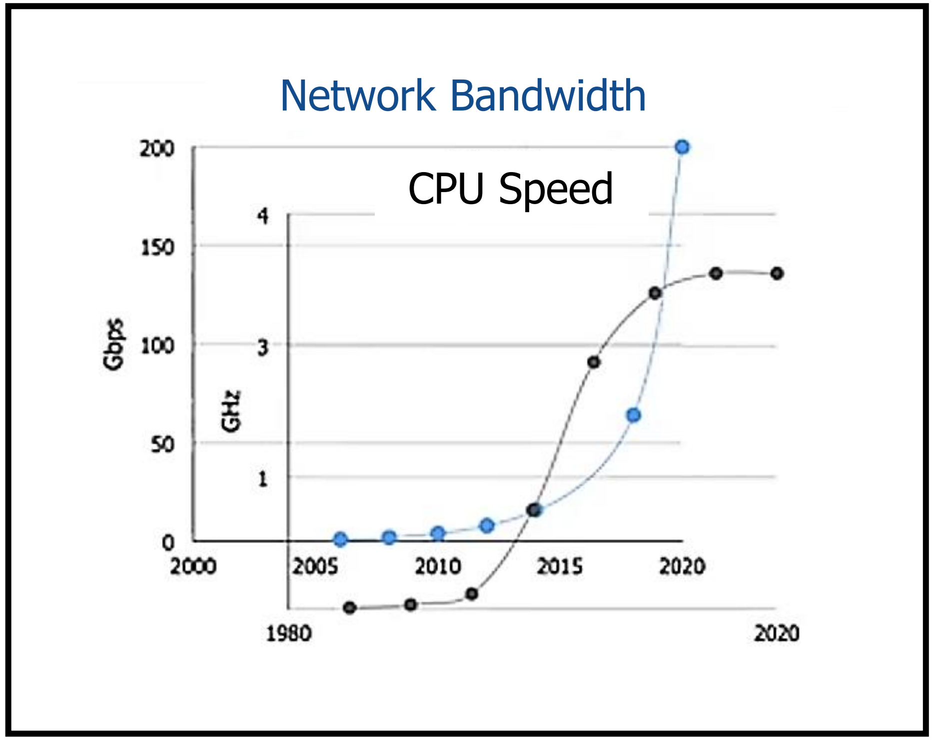
yj0225@dankook.ac.kr

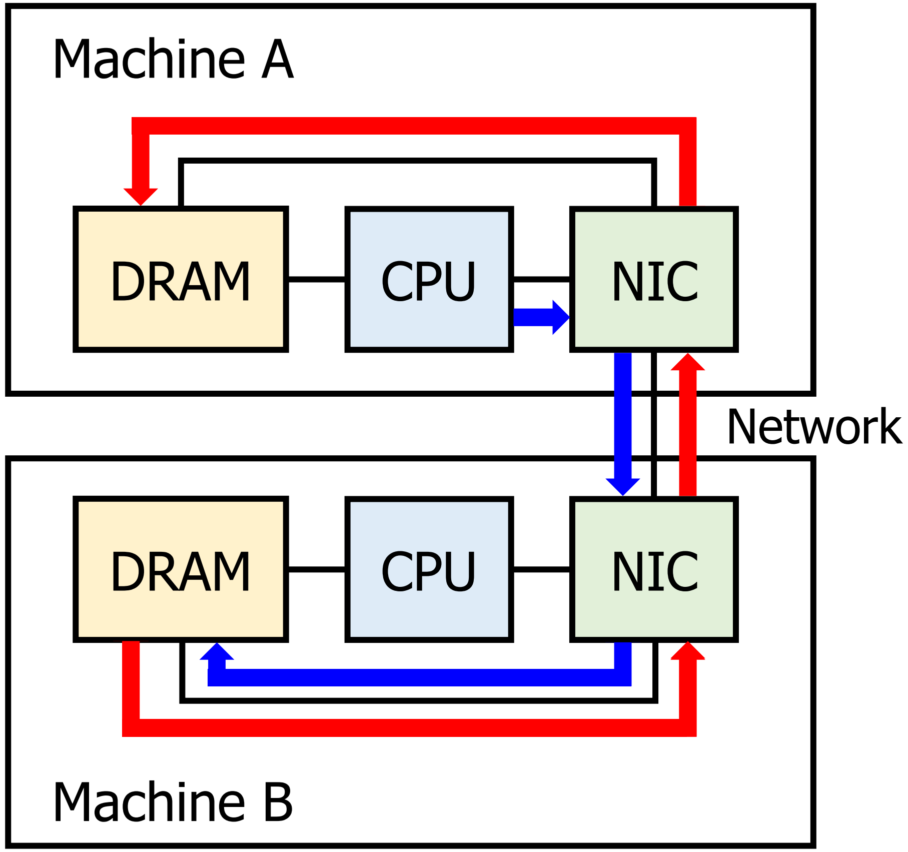
Contents

1. Introduction
2. Background
3. Motivation
4. PRISM
5. Evaluation
6. Conclusion

Remote Direct Memory Access (RDMA)

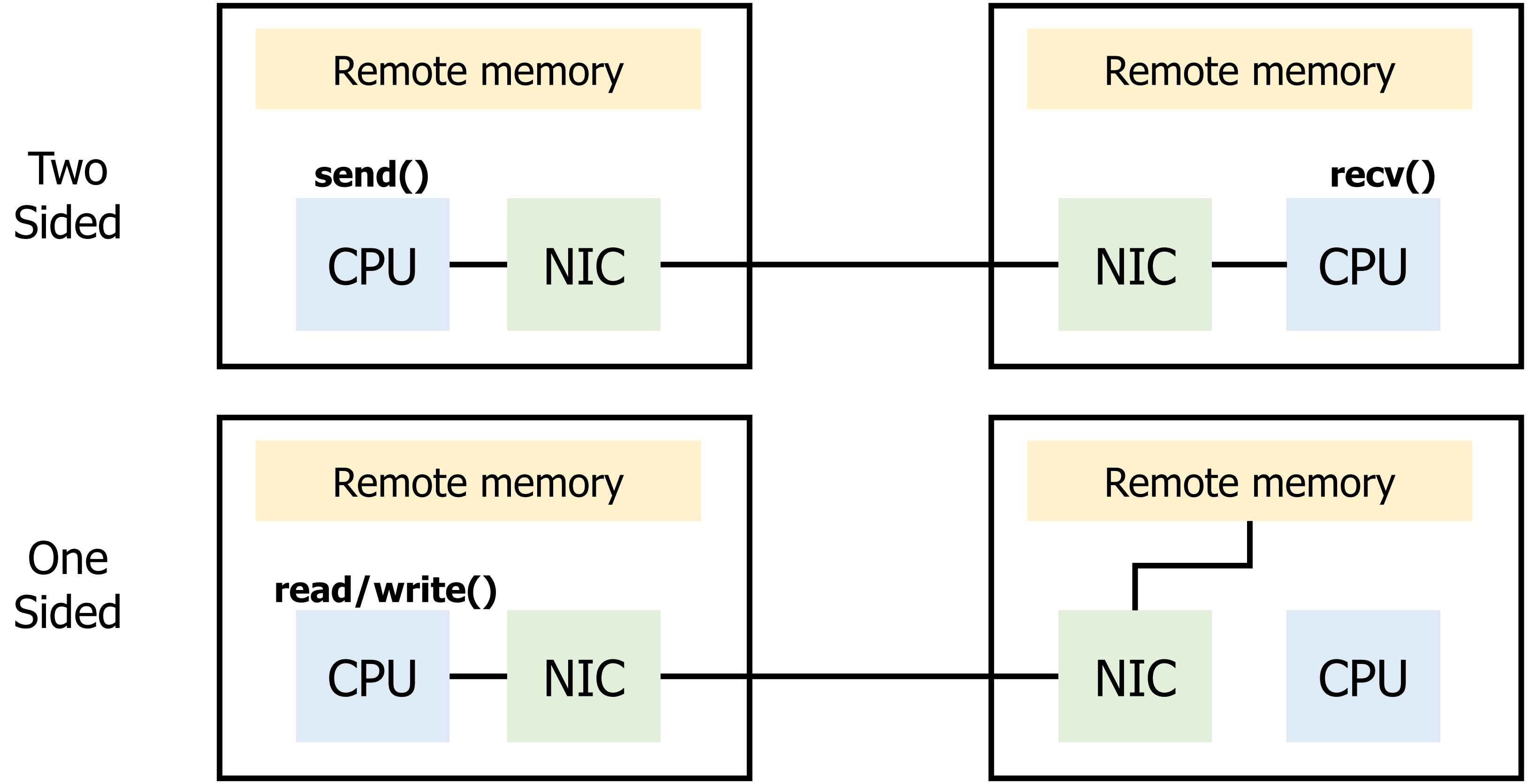
- Network bandwidth increases relative to CPU speed
- Kernel bypassing, CPU Offloading technology





RDMA provides two types of operations

- Two-sided / One-sided operations



- Less CPU Efficient
- Generalizable Interface

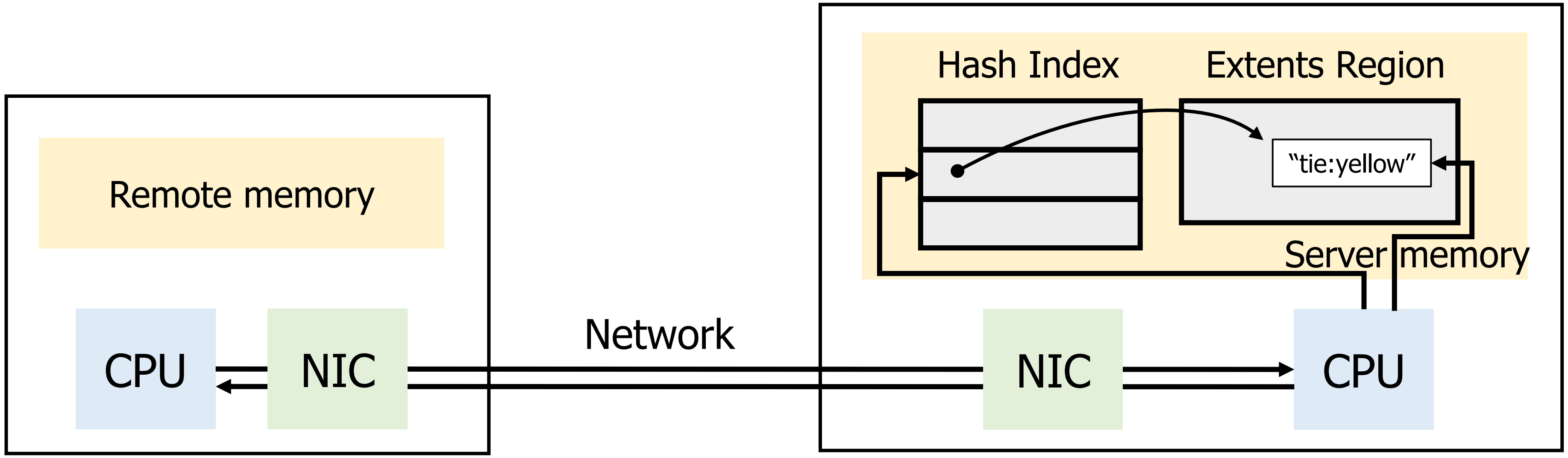
- More CPU Efficient
- Restrictive Interface

Embedded System Lab.

Indirect reads: One-sided vs. Two-sided

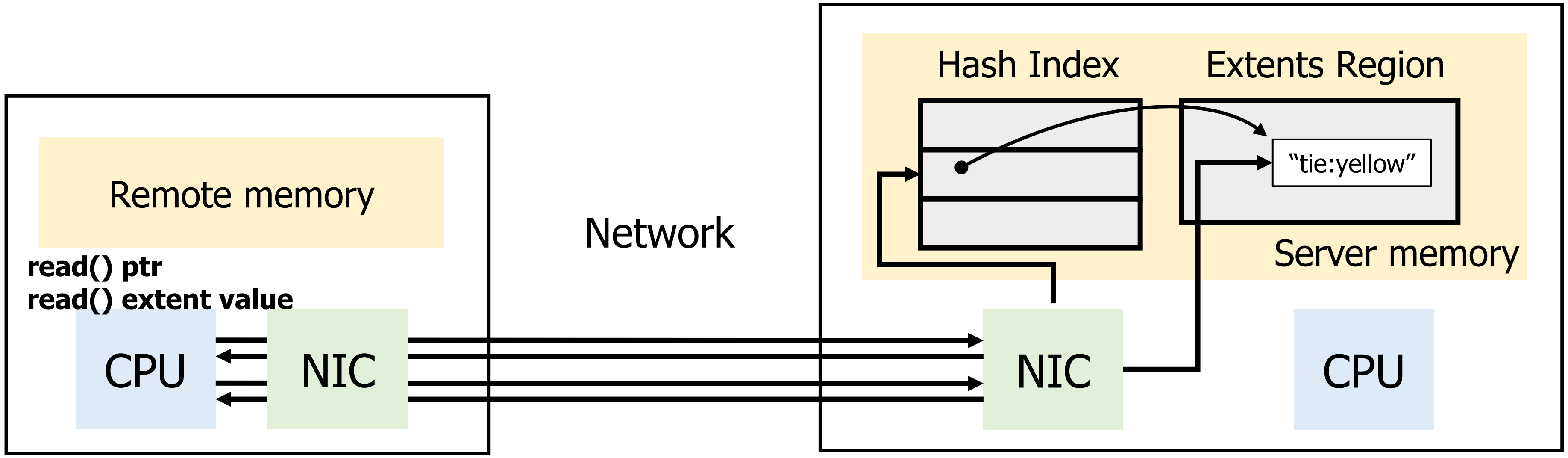
Two Sided

- **Involves CPU**
- **1 Roundtrip**



One Sided

- **No CPU involved**
- **2 Roundtrips**



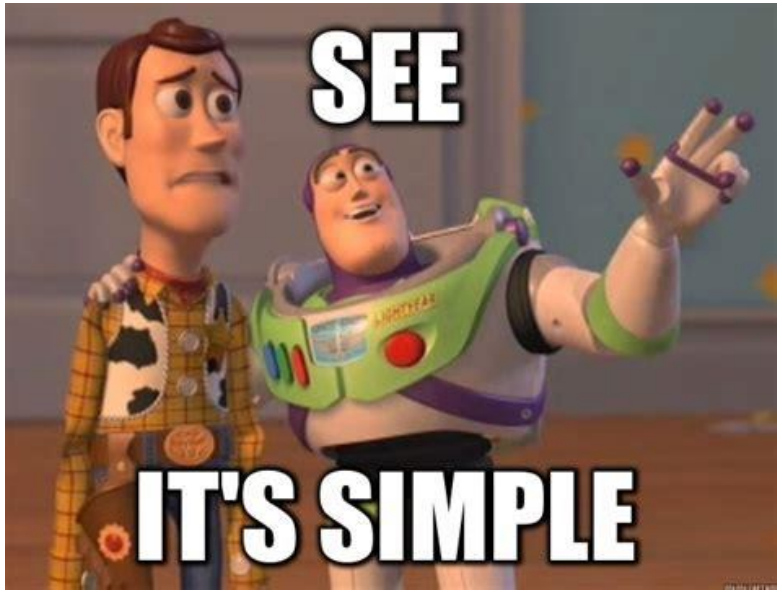
Difficulty of adapting applications to run on RDMA

- Applications are limited to the current RDMA read/write interface

Extend the RDMA interface

PRISM's API design principles

- Generality
- Minimal interface complexity
- Minimal implementation complexity



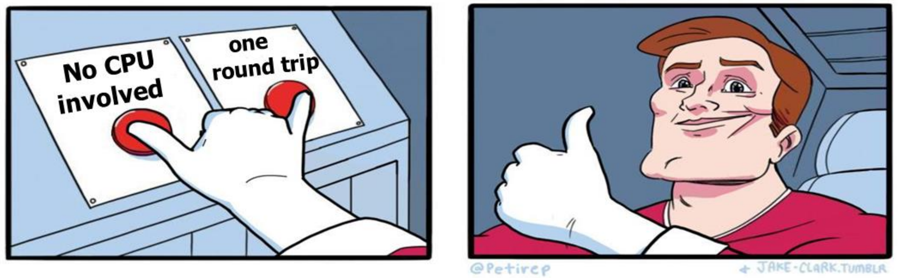
PRISM Primitives

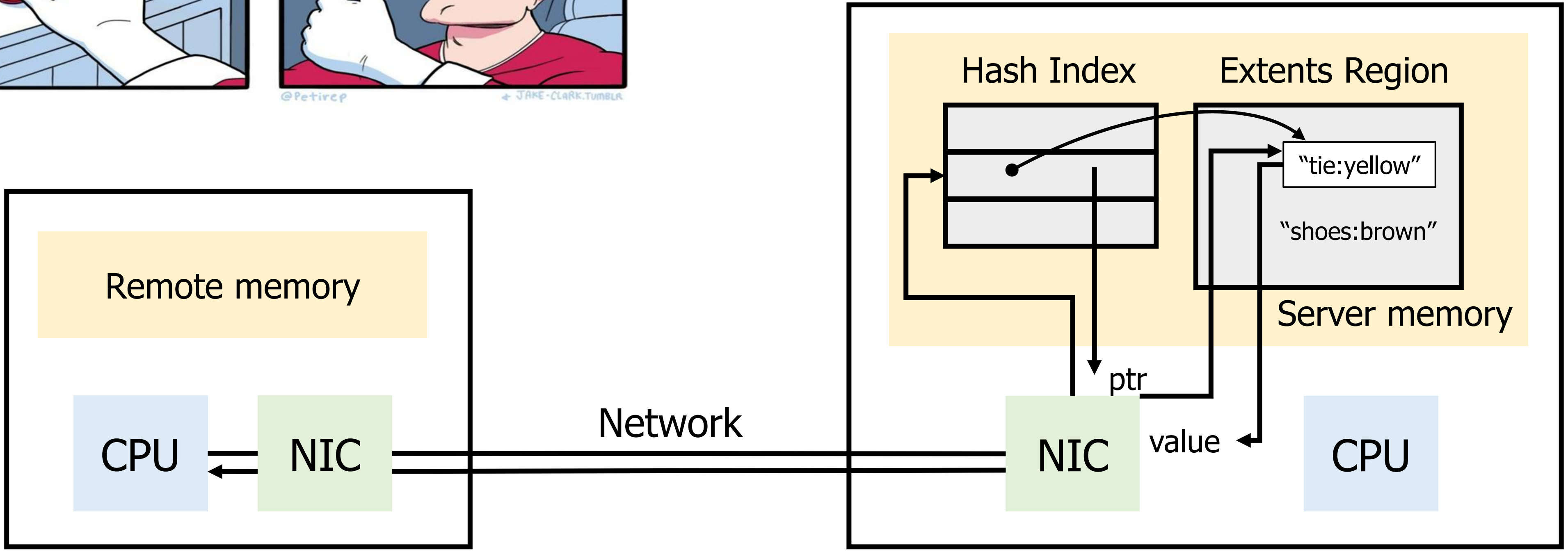
- Indirect, Enhanced CAS, Allocation, Operation Chaining

Indirect Reads/Writes	The target address specified by the operation can instead be interpreted as a pointer to the actual target
Enhanced Compare And Swap	Extends RDMA CAS to provide support for arithmetic comparisons (>, <) during the compare phase
Allocation	Allows memory allocation on the data-plane from a pre-registered pool of memory
Operation chaining	Allows for the execution of a chain of other PRISM primitives at the NIC

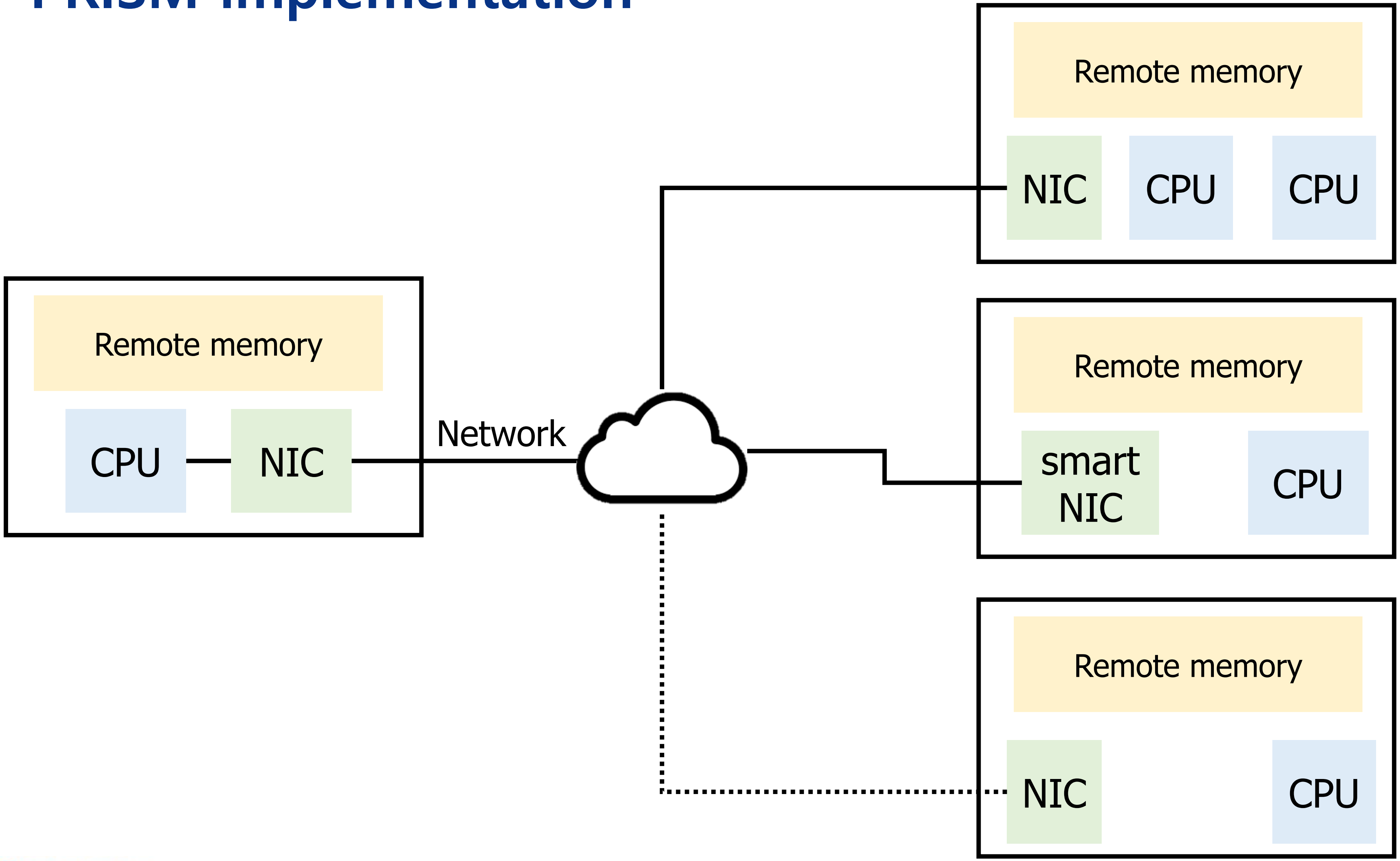
Indirect Reads with PRISM

- No CPU involved
- 1 Network Roundtrip





PRISM implementation



Snap inspired Software RDMA Prototype

Mellanox BlueField-1 SmartNIC Prototype

PRISM API could be added to a NIC ASIC

Applications designed with PRISM

- **PRISM-KV**: a Key-Value Store that implements both read and write operations using the One Sided PRISM API
- **PRISM-RS**: a replicated storage system that implements the ABD quorum replication protocol
- **PRISM-TX**: a transactional storage system that implements a timestamp-based optimistic concurrency control protocol using PRISM's primitives.

- **PRISM-KV: Key-Value Storage**

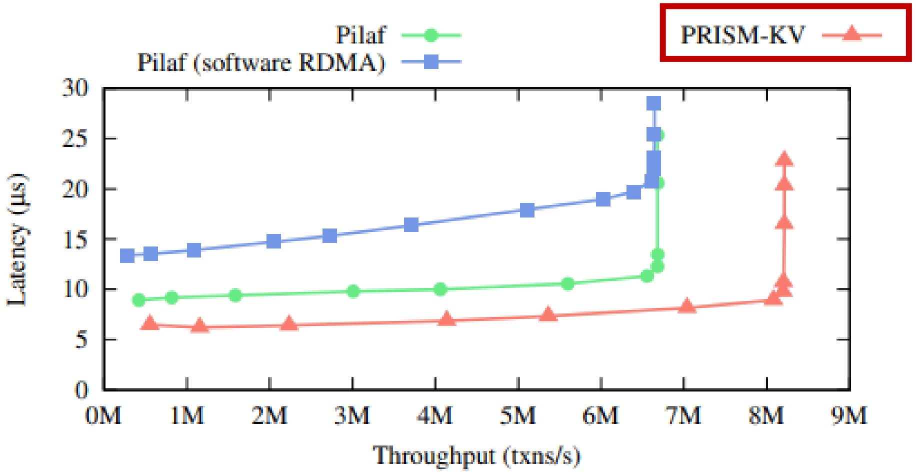


Figure 3. Throughput versus average latency comparison for PRISM-KV and Pilaf, 100% reads, uniform distribution.

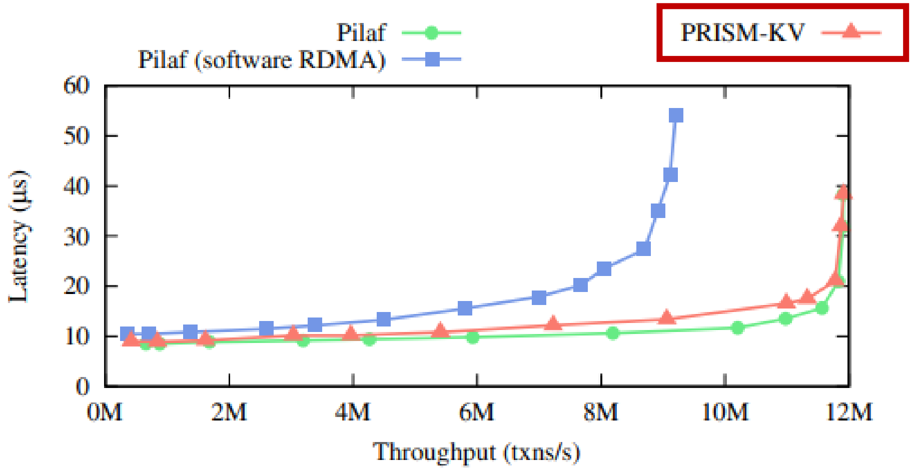


Figure 4. Throughput versus average latency for PRISM-KV and Pilaf, 50% reads, uniform distribution.

- **Latency difference is about 2X**
- **22% higher read throughput**

matches RDMA-enabled Pilaf for 50/50 mixed workload

- **PRISM-RS: Replicated Block Store**

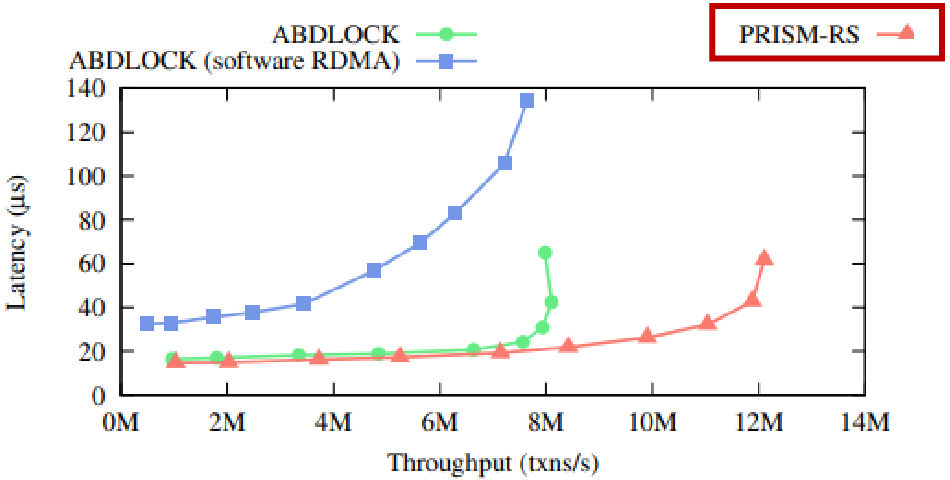


Figure 6. Throughput-latency comparison between PRISM-RS and the two variants of lock-based ABD.

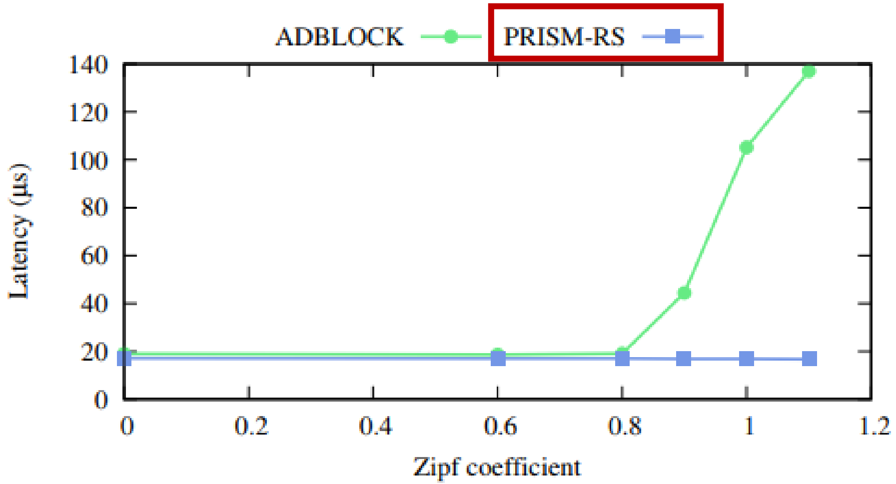


Figure 7. Latency comparison between PRISM-RS and ABD-LOCK for various degrees of contention.

- **2µs faster than ABD-LOCK**
- **~4million more ops/sec**

Dramatic benefits where there is contention on popular keys

- **PRISM-TX: Distributed Transactions**

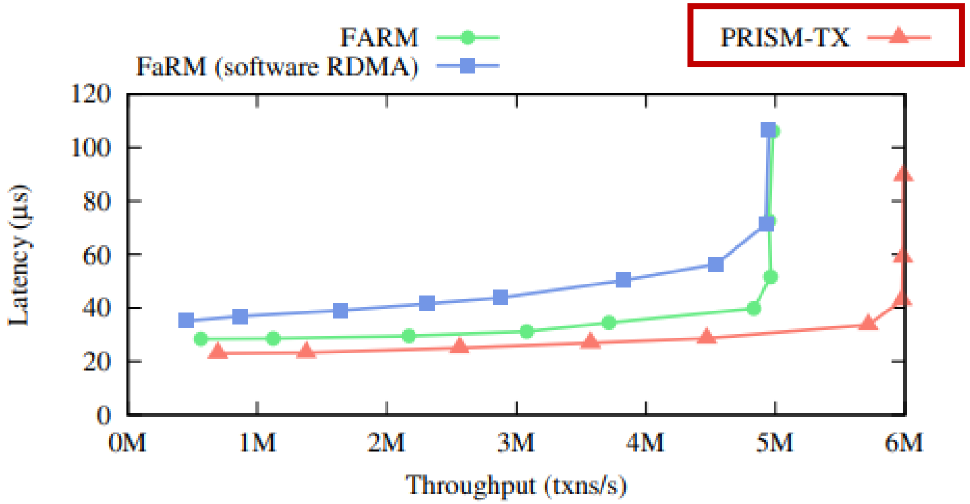


Figure 9. Throughput-latency comparison between PRISM-TX and FaRM for YCSB-T workload with low contention.

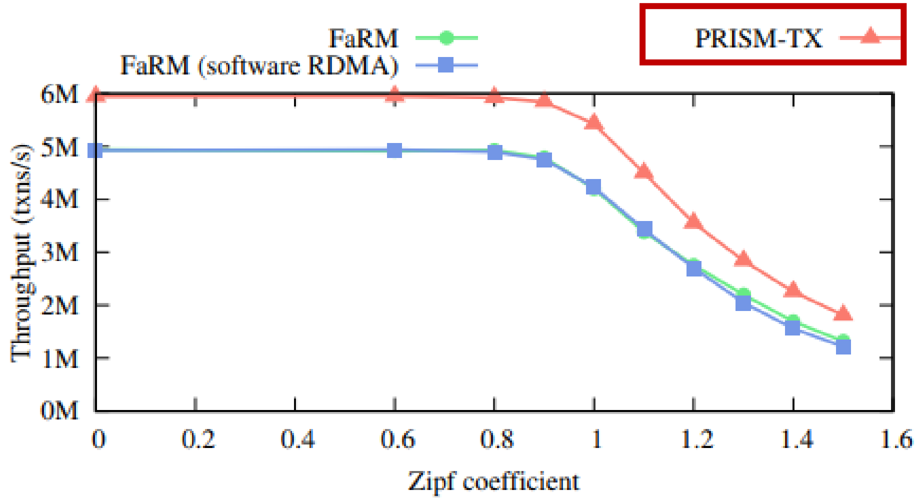


Figure 10. Peak throughput comparison between PRISM-TX and FaRM for YCSB-T workload with varying contention.

- **5.5µs faster than FaRM**
- **1 million more txns/s**

Maintains performance benefit under high contention

PRISM

- The current RDMA Interface isn't expressive enough to benefit most distributed systems applications

- PRISM proposes a set of generic primitives that extend the RDMA API

  : Indirect, enhanced CAS, allocation, operation chanining

- Demonstrate the PRISM API's benefits by designing 3 new applications

  : PRISM-KV, PRISM-RS, PRISM-TX

PRISM: Rethinking the RDMA Interface for Distributed Systems

Matthew Burke, Sowmya Dharaniparagada, Shannon Joyner, Adriana Szekeres, Jacob Nelson, Irene Zhang, Dan R. K. Ports

SOSP'21

Thank You!

2022. 04. 12

Presentation by Han, Yejin

yj0225@dankook.ac.kr