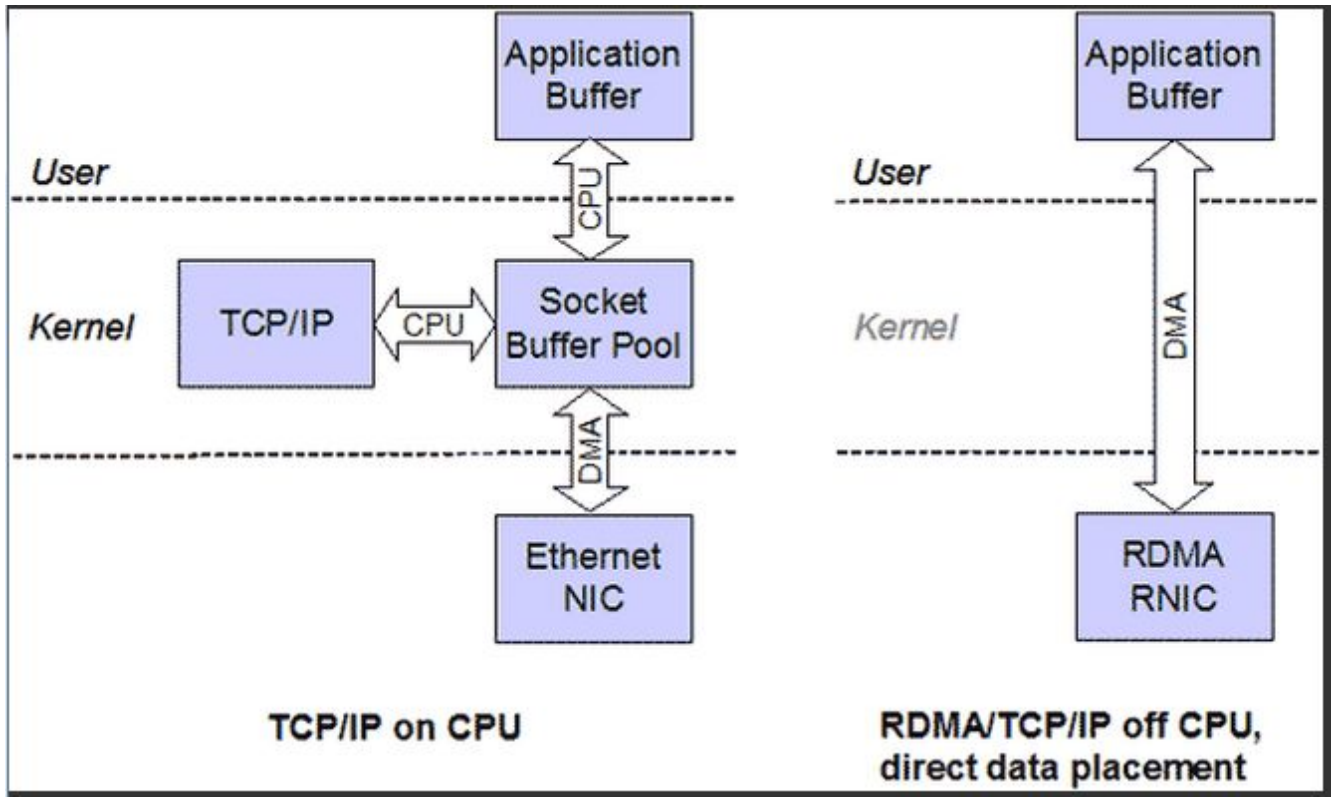
# Using RDMA Efficiently for Key-Value Services

SIGCOMM2014

# Brief Introduction to RDMA

- RDMA is the communication method used in computation clusters.
- It alows the host to directly work on memory of a remote host without notifying the remote host.
- RDMA has the smallest latency, only 1-3us.
- Now the price of the RDMA adapters is comparable to that of ethernet adapaters, driving datacenter operators to equipt datacenters with this high speed tool.

# Brief Introduction to RDMA



# Basic RDMA Semantics

- Verbs: Functions that userspace program uses to access RNIC (RDMA NIC).

- Queue Pair: There is a send queue in the sender and receive in the receiver, forming a queue pair. Each queue has a completion queue.

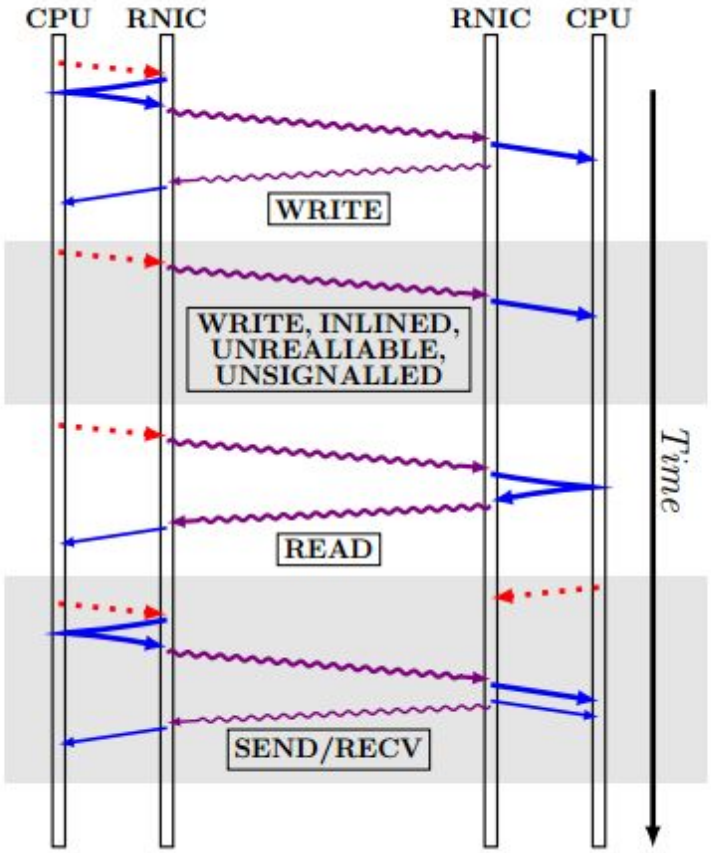
- Memory Semantics: RDMA READ and RDMA WRITE. They work directly on the remote memory without notifying the remote host's kernel

- Channel Semantics: RDMA SEND and RDMA RECV. SEND's payload is written to a remote memory address that is specified by the responder in a pre-posted RECV.

# Basic RDMA Semantics

- RDMA transport types:
  - Connected transport: Require a connection between a queue pair. Support both reliable and unreliable connections. In unreliable connections, there's no ACK of the received packets.
  - Unconnected transport: A sender queue can communicate with many other queues.

# Basic RDMA Semantics




| Verb | RC | UC | UD |
|---|---|---|---|
| SEND/RECV | ✓ | ✓ | ✓ |
| WRITE | ✓ | ✓ | ✗ |
| READ | ✓ | ✗ | ✗ |

# Choosing the Correct Semantic Combination

- To implement a key-value store, how to choose the semantic combination?

- Pilaf: For Get, use pure RDMA SEND. For Put, use SEND.

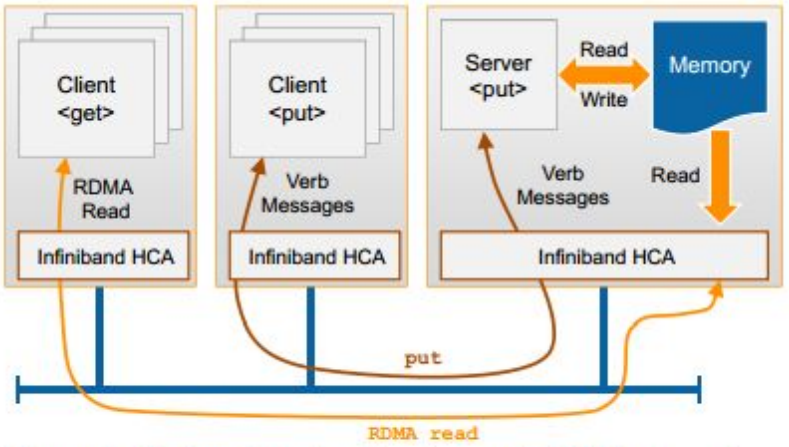
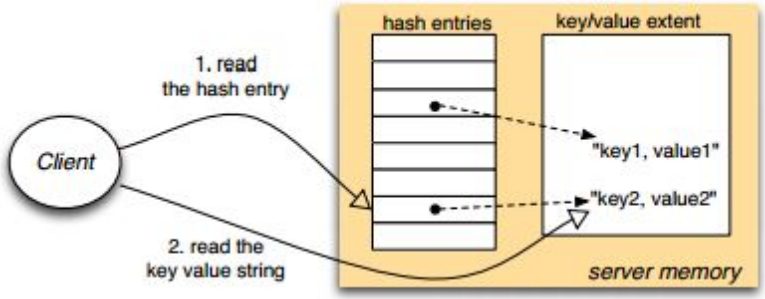



**Figure 3:** Pilaf restricts the clients' use of RDMAs to read-only `get` operations and handles all `puts` at the server using verb messages.



# Problem with Pilaf

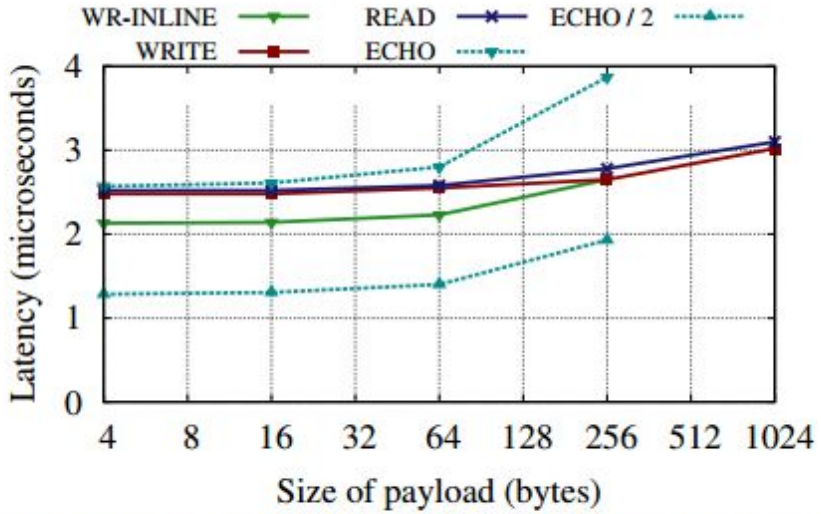
- A single Get operation needs several READ to traverse the hash table data structure.
- Each READ requres a round trip time to finish.
- A Get operation requires 2.6 READ on everage. (In this paper, a Get operation is finished within one round trip time.)
- Server is unaware of the READ. Pilaf use "self-verifying" data-structure to provide concurrent read-write access to the memory. (Use a checksome to protect the memory area, an invalid check indicates a invalid read.). This design can not use memory efficiently.

# HERD's Design

- Use RDMA WRITE over an Unreliable Connection to place Put/Get request directly on the server.
  - WRITE over an Unreliable Connection only requires half of a round trip time.
  - A Get implemented using READ need to READ multiple times and each READ needs one round trip time.
  - Latency is greatly reduced.

- Use SEND over Unreliable Datagram to place the response back to the sender
  - SEND over Unreliable Datagram also only requires half of a round trip time.
  - More over, for a multi-client to one server key value store system, SEND over UD scales well as the number of client increases.
  - This is because SEND over UD supports one-to-many communication pattern, only need one send queue at the server and one receive queue at each receiver.

# HERD's Design




**(b)** The one-way latency of WRITE is half of the ECHO latency. ECHO operations used unsignaled verbs.

# HERD's Design

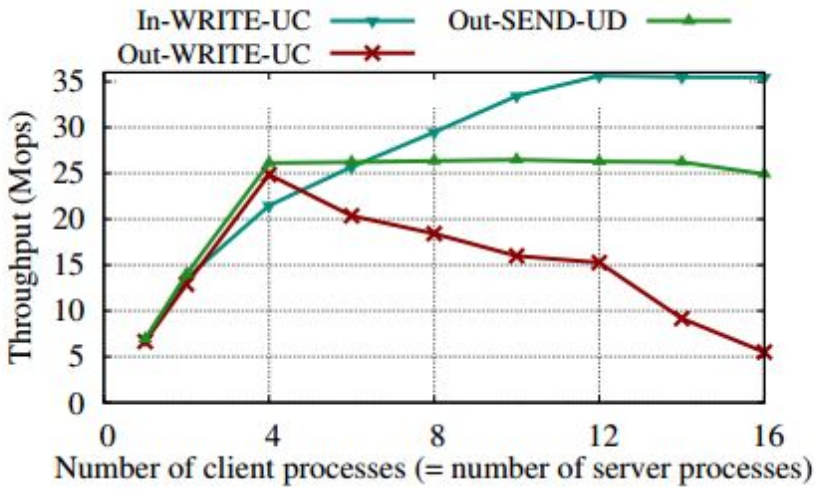



**Figure 6: Comparison of UD and UC for all-to-all communication with 32 byte payloads.** Inbound WRITEs over UC and outbound SENDs over UD scale well up to 256 queue pairs. Outbound WRITEs over UC scale poorly. All operations are inlined and unsignaled.

# Requests

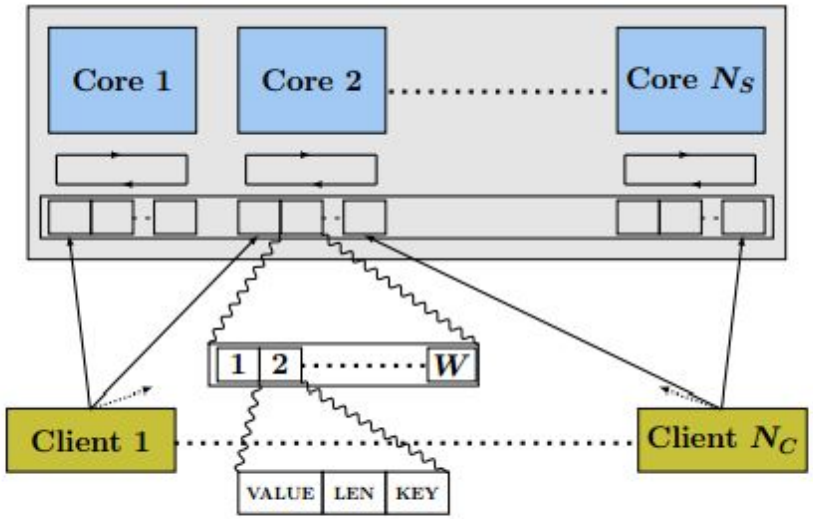



**Figure 8: Layout of the request region at the server**

# Response

- Each server process has a UD queue for holding the response.

- Each client has Ns UD queues for receiving the responses from each server processes.

- Request/response level transmission control:

  After the client posts W requests, it polls for the completion event of the server response. For each completion event, the client posts a new request.

# Performance

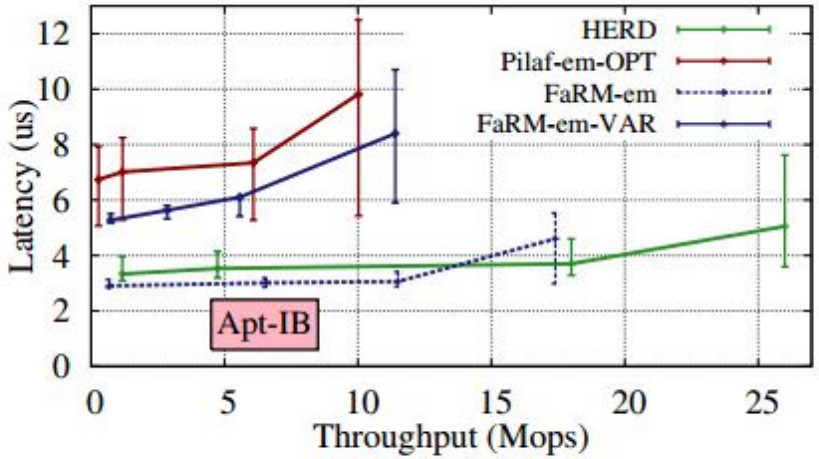



**Figure 11: End-to-end latency with 48 byte items and read-intensive workload**

# Performance

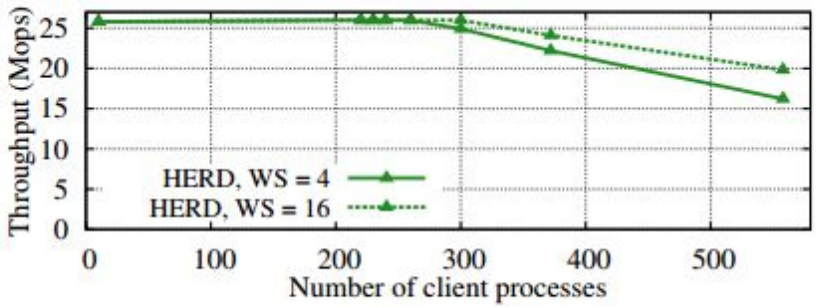



**Figure 12: Throughput with variable number of client processes and different window sizes**

# Performance

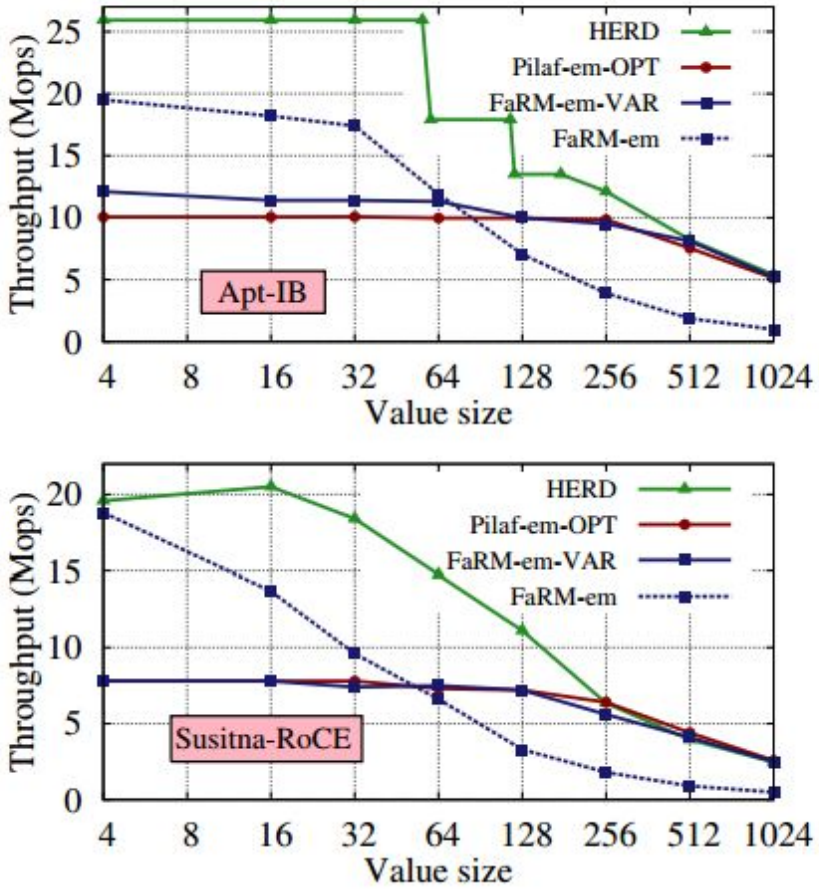



**Figure 10: End-to-end throughput comparison with different value sizes**

# Summary

- Target in a specific scenario. A single server multiple client key-value store system, with small key-value size.
- Remove additional network overhead, each operation finishes within one round trip time.
- Offload some work of the client to the server.(A client using READ based Get needs to query the server memory several times.)
- They have already implemented a very fast key-value store system called MICA. So they only focus on how to provide a faster interface to this storage system.