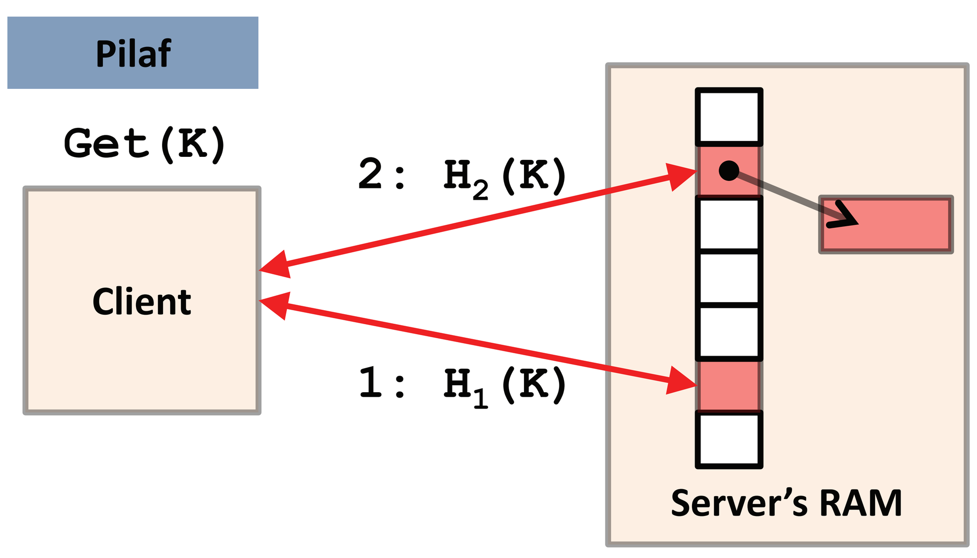
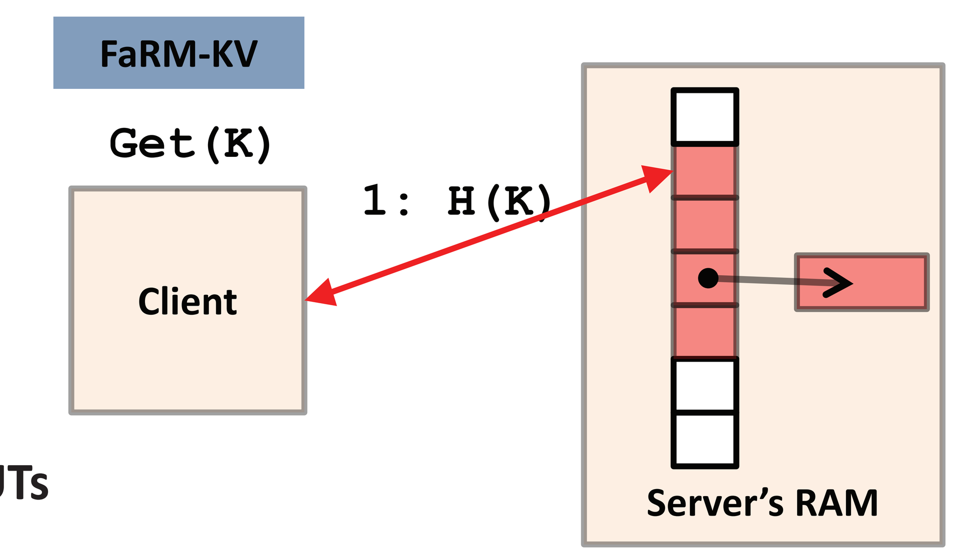
Using RDMA Efficiently for Key-Value Services

Anuj Kalia (CMU), Michael Kaminsky (Intel Labs), David G. Andersen (CMU)

TODAY'S SYSTEMS USE > 1 RDMA READ TO ACCESS REMOTE HASH-TABLES



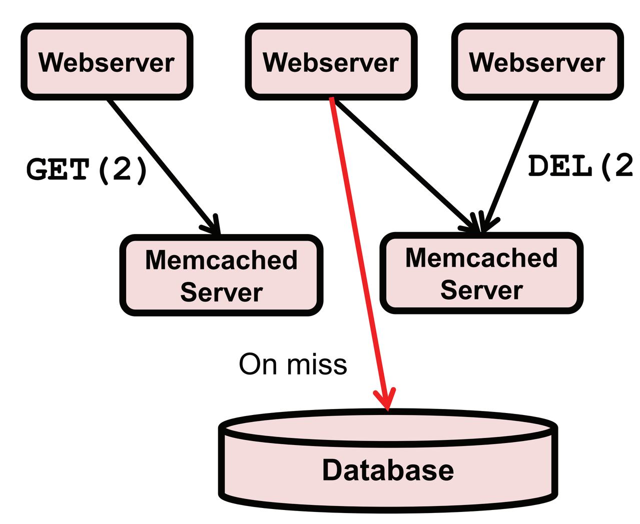
Pilaf [1]
- Cuckoo hashing
- 2.5 RDMA reads per GET
- GET throughput: $T_{READ}/2.5$
- GET latency: $L_{READ} * 2.5$
- RDMA verb messages for PUTs



FaRM-KV [2]
- Hopscotch hashing
- 1 or 2 RDMA reads per GET
- GET throughput and latency depend on bucket size
- RDMA writes for PUTs

BACKGROUND

In-memory key-value services
- Interface: GET(key), PUT(key, value), DELETE(key)
- Data is stored in RAM
  - A key is mapped to a pointer using an index (hash table, tree). Value is stored at the pointer
- Examples: Memcached, Redis, RAMCloud



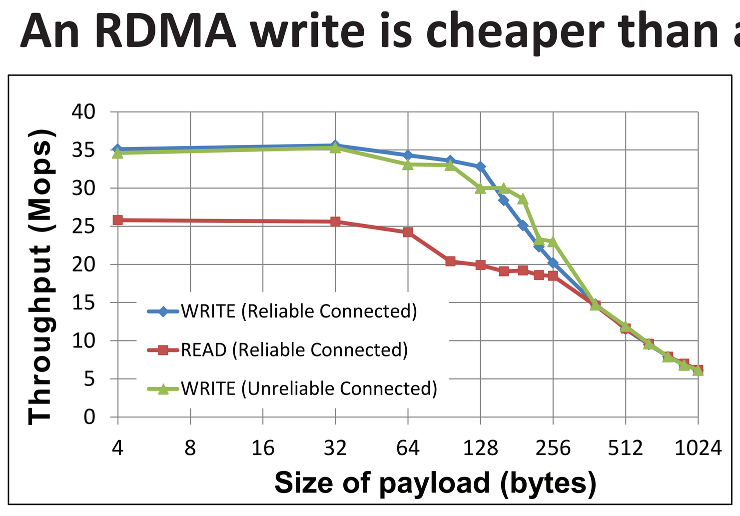
RDMA
- Low latency: (2-3 µs RTT) vs 30-60 µs for Ethernet
- Memory verbs: direct access to memory of remote host
  - `READ(local_buf, size, remote_buf)`
  - `WRITE(local_buf, size, remote_buf)`
- Messaging verbs:
  - `SEND(local_buf, size)`
  - `RECV(local_buf, size)`

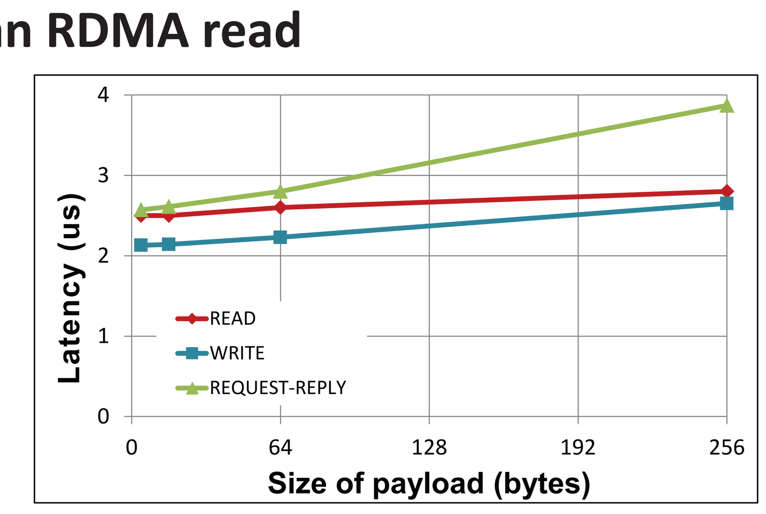
OUR APPROACH: DON'T PAY MICROSECONDS TO SAVE NANOSECONDS

Let the server traverse the data structure
- Memory access latency (~ 100 ns) << RDMA read latency (~ 2-3 µs)

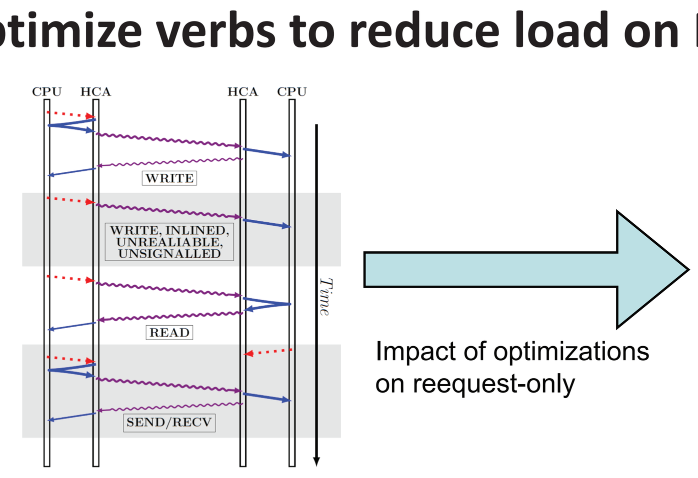
Requirement: A fast and scalable request-reply mechanism

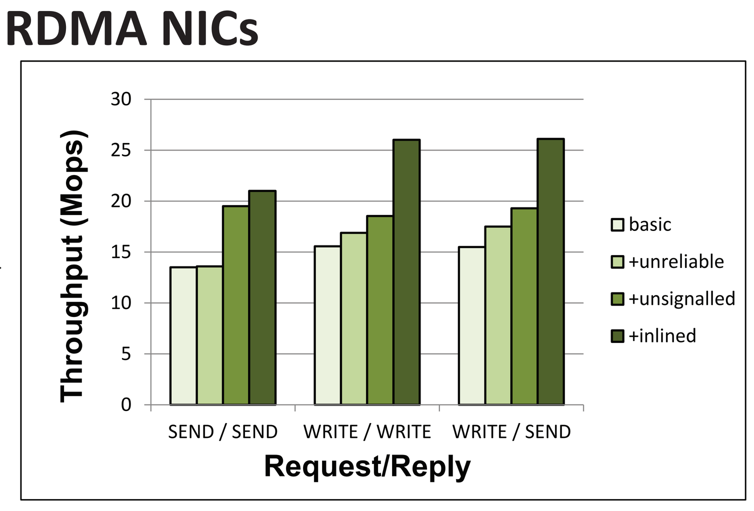
Core ideas:
- An RDMA write is cheaper than an RDMA read





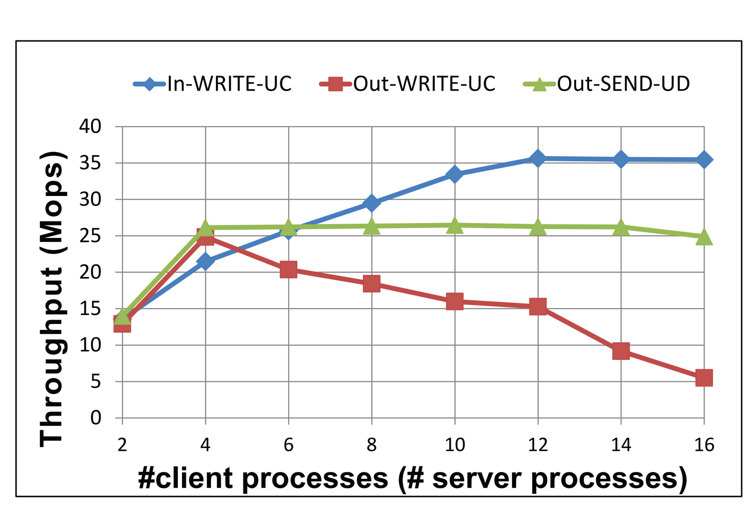
- Optimize verbs to reduce load on RDMA NICs





- Only inbound RDMAs (requests) scale with the number of clients → use datagram transport for replies.

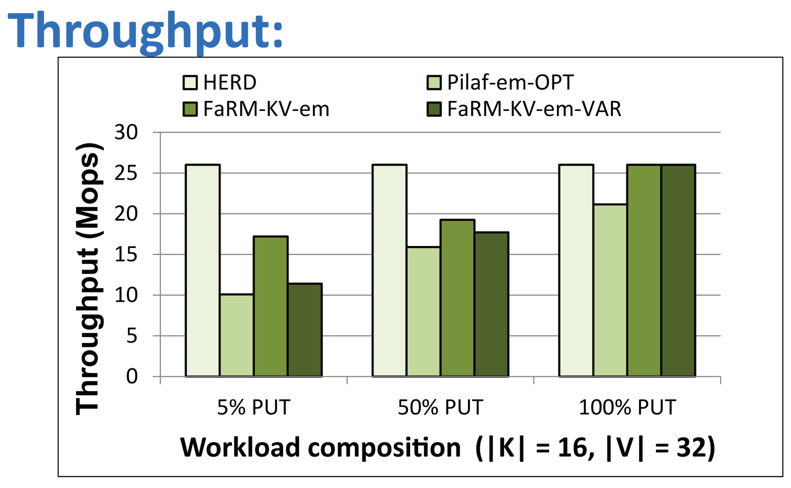
Messaging verbs for scalability:
- Messaging is expensive only at the receiver's side: HERD's server does not post RECVs
  - RECV cost amortized over clients
- Previous assumption: Messaging verbs are more expensive than 2 READs
  - With our optimizations, this is not true

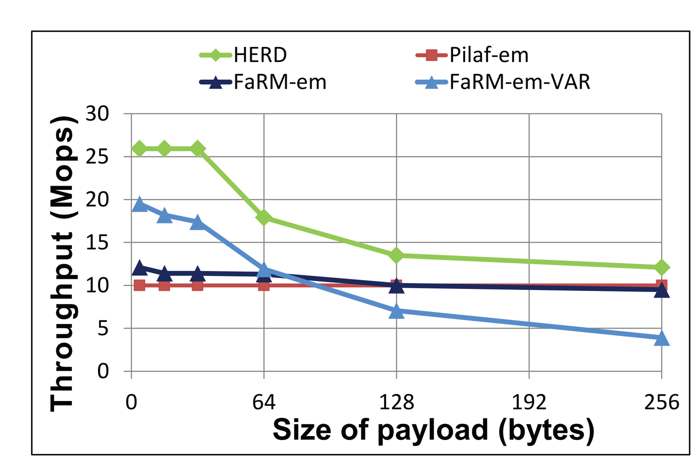


Design:
- Borrow lossy index and circular log data structures from MICA[3]
- Clients write requests to the appropriate server core using RDMA writes
- Server computes response and replies with a SEND message over a datagram connection

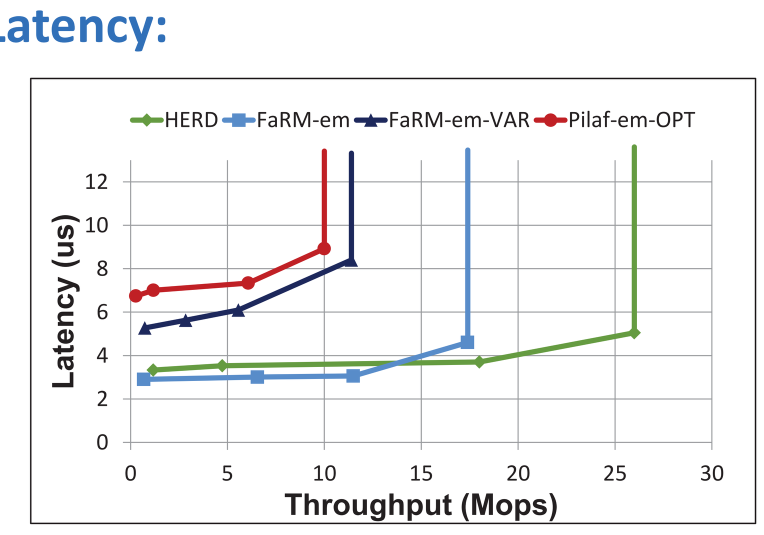
Evaluation: Comparison against stripped versions of Pilaf and FaRM-KV

Throughput:





Latency:



- HERD delivers 26 Mops with 5 µs average latency
- Over 2X higher throughput than Pilaf and FaRM (with variable length keys)
- Average latency over 2X lower than Pilaf's and FaRM-KV's at their peak throughput

Paper and code:
Using RDMA Efficiently for Key-Value Services (Anuj Kalia, Michael Kaminsky, David G. Andersen), SIGCOMM 2014. https://github.com/efficient/HERD

[1] Using One-Sided RDMA Reads to Build a Fast, CPU-Efficient Key-Value Store. (Christopher Mitchell, Yifeng Geng, Jinyang Li), ATC 2013
[2] FaRM: Fast Remote Memory (Aleksandar Dragojević, Dushyanth Narayanan, Miguel Castro) NSDI 2014
[3] MICA: A Holistic Approach to Fast In-Memory Key-Value Storage (Hyeontaek Lim, Dongsu Han, David G. Andersen, Michael Kaminsky) NSDI 2014

Intel Science & Technology Center for Cloud Computing
Carnegie Mellon University · Georgia Tech · intel · Princeton University · UC Berkeley · University of Washington