HIGH-PERFORMANCE NETWORKING
:: USER-LEVEL NETWORKING
:: REMOTE DIRECT MEMORY ACCESS

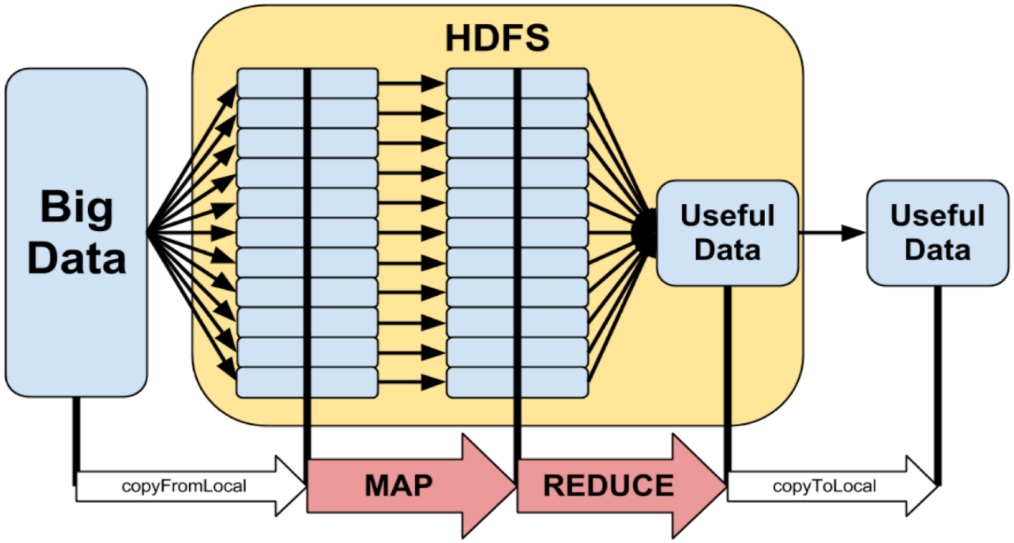
CS6410

Jaeyong Sung (Nov 20, 2014)

Part 2

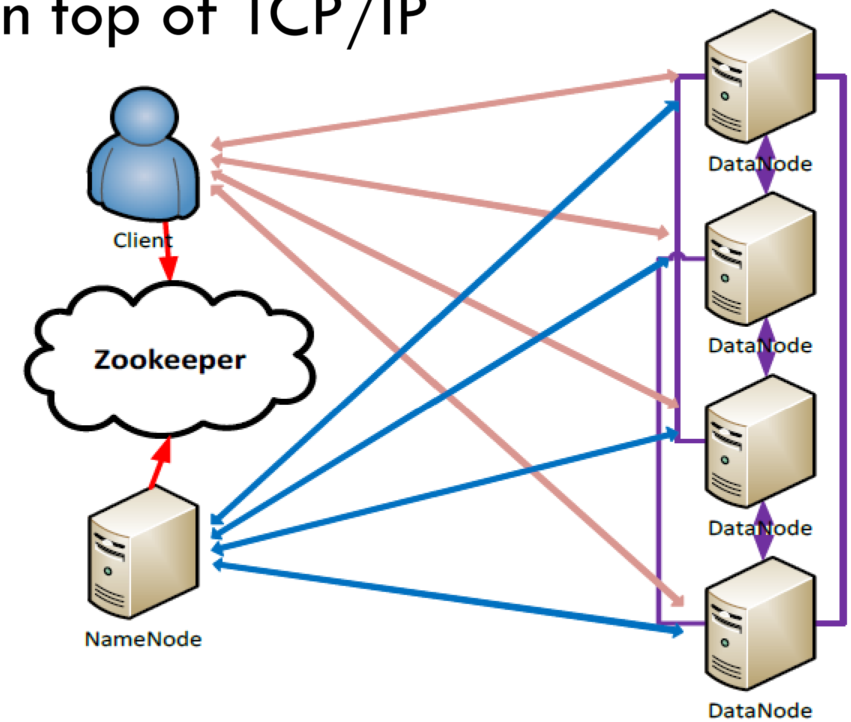
Hadoop

- Big Data
  - very common in industries
    - e.g. Facebook, Google, Amazon, …
- Hadoop
  - open source MapReduce for handling large data
  - require lots of data transfers



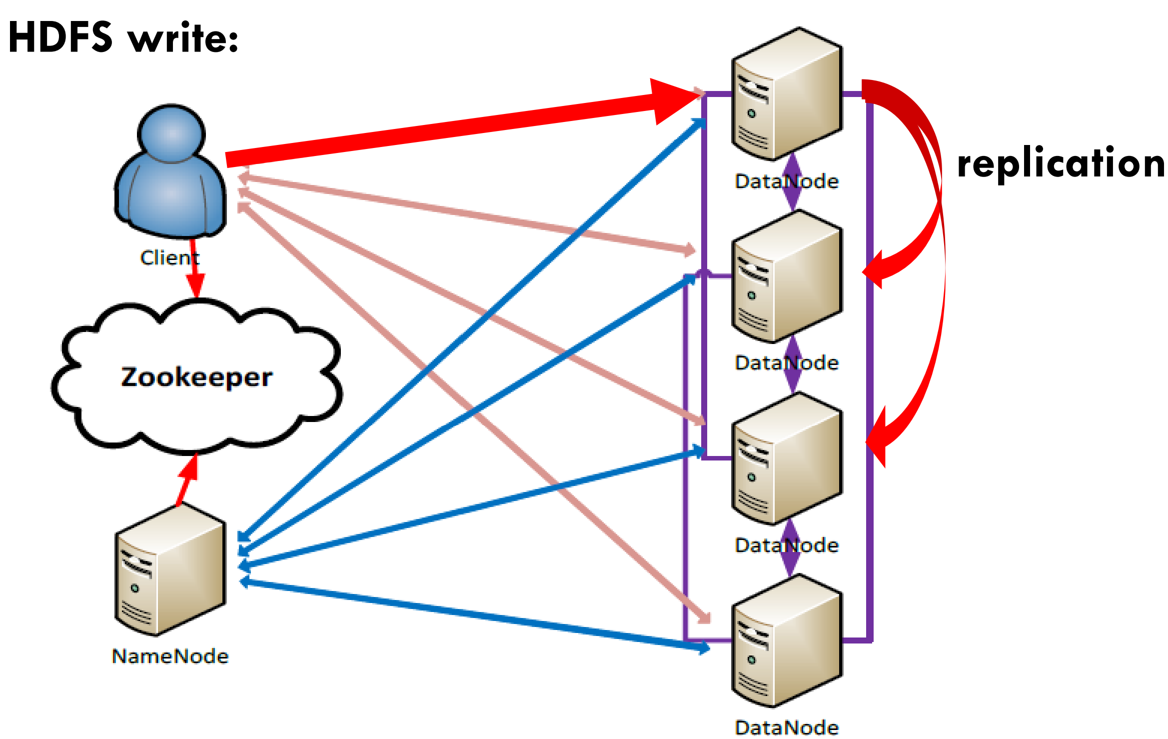
Hadoop Distributed File System

- primary storage for Hadoop clusters
  - both Hadoop MapReduce and HBase rely on it
- communication intensive middleware
  - layered on top of TCP/IP



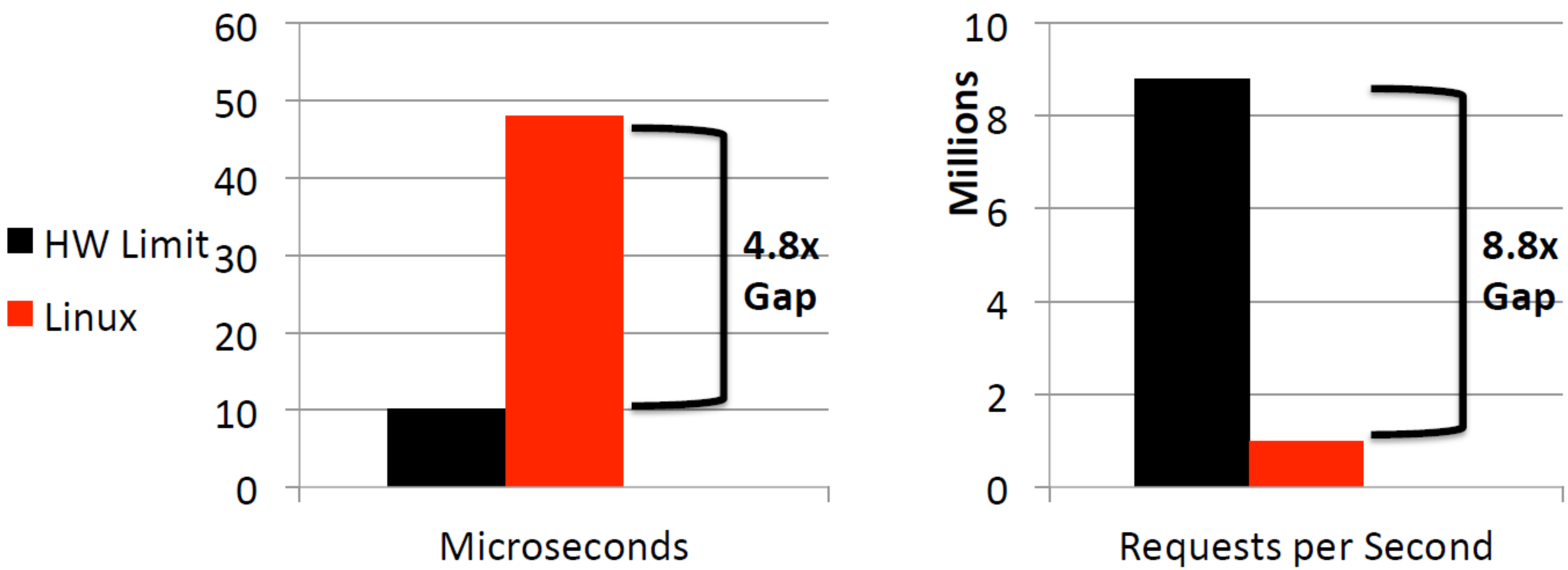
Hadoop Distributed File System

- highly reliable fault-tolerant replications
- in data-intensive applications, network performance becomes key component



Software Bottleneck

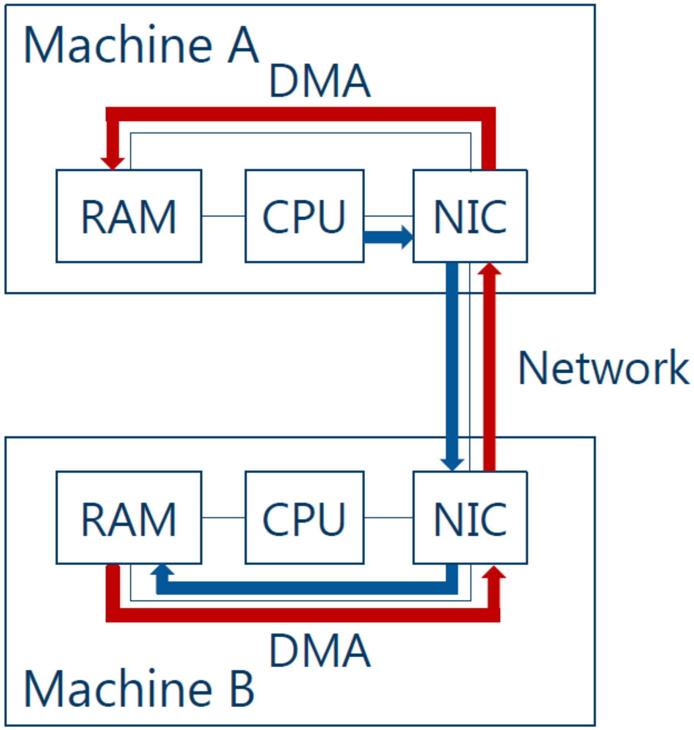
- Using TCP/IP on Linux,
  - TCP echo



RDMA for HDFS

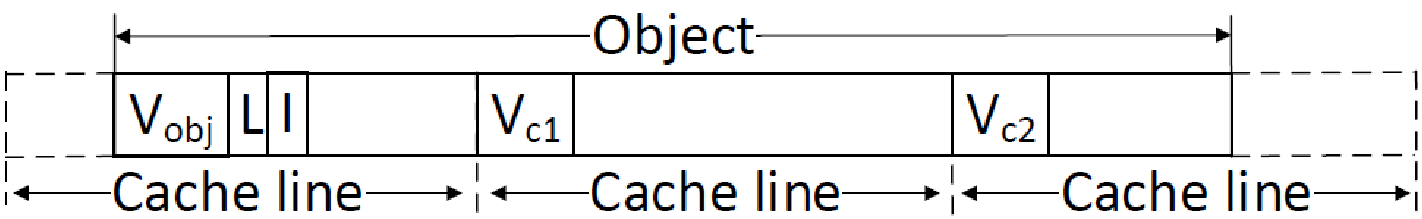
- data structure for Hadoop
  - <key, value> pairs
  - stored in data blocks of HDFS

- Both write(replication) and read can take advantage of RDMA



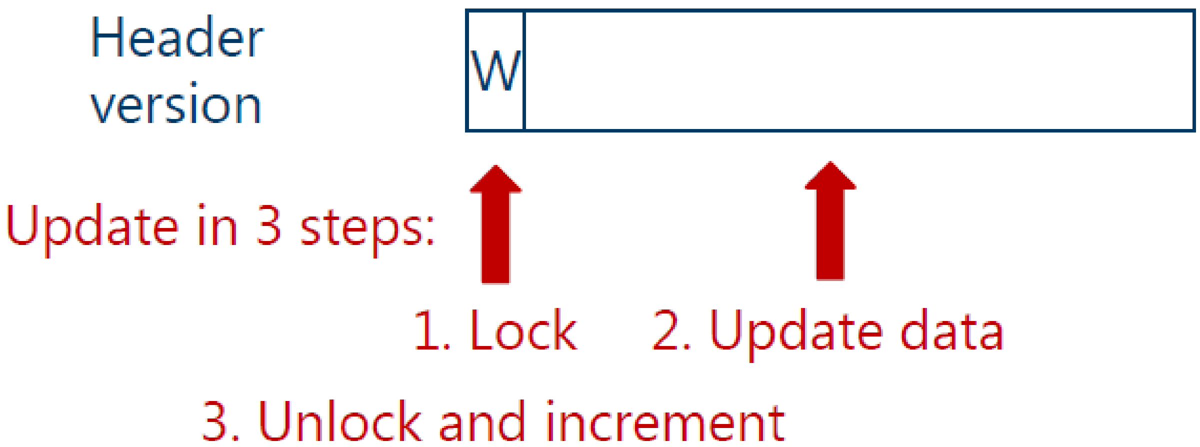
FaRM: Fast Remote Memory

- relies on cache coherent DMA
  - object's version number
    - stored both in the first word of the object header and at the start of each cache line
    - NOT visible to the application (e.g. HDFS)



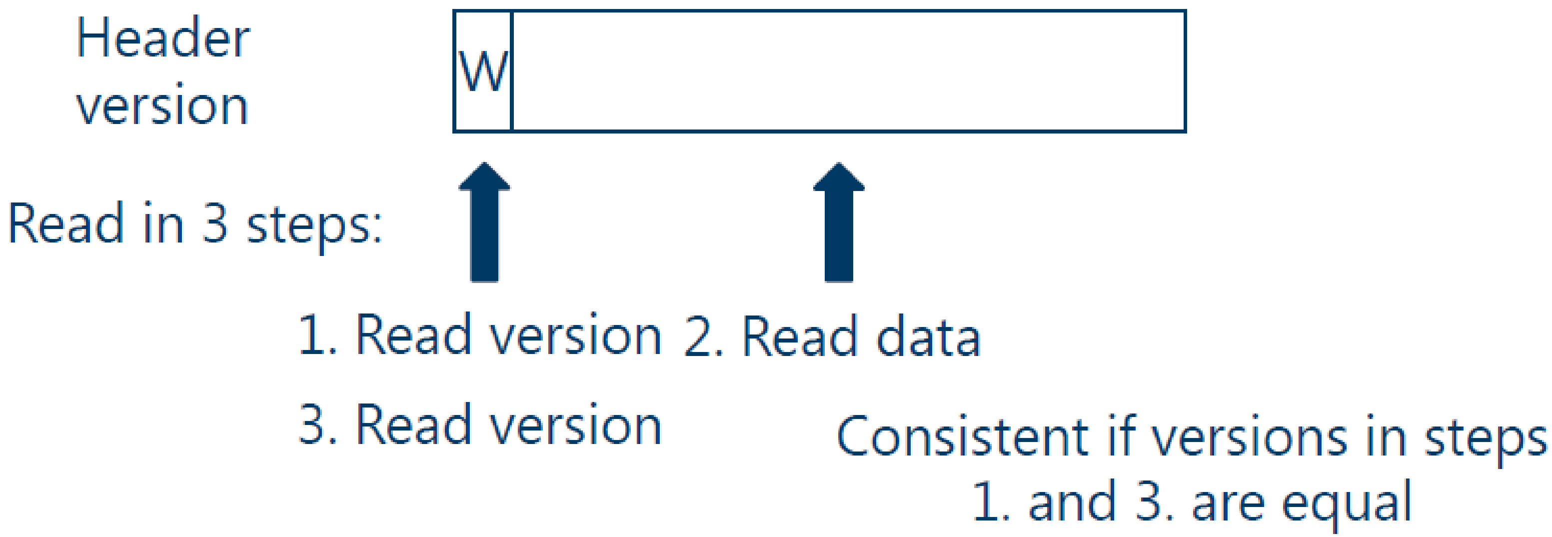
Traditional Lock-free reads

- For updating the data,



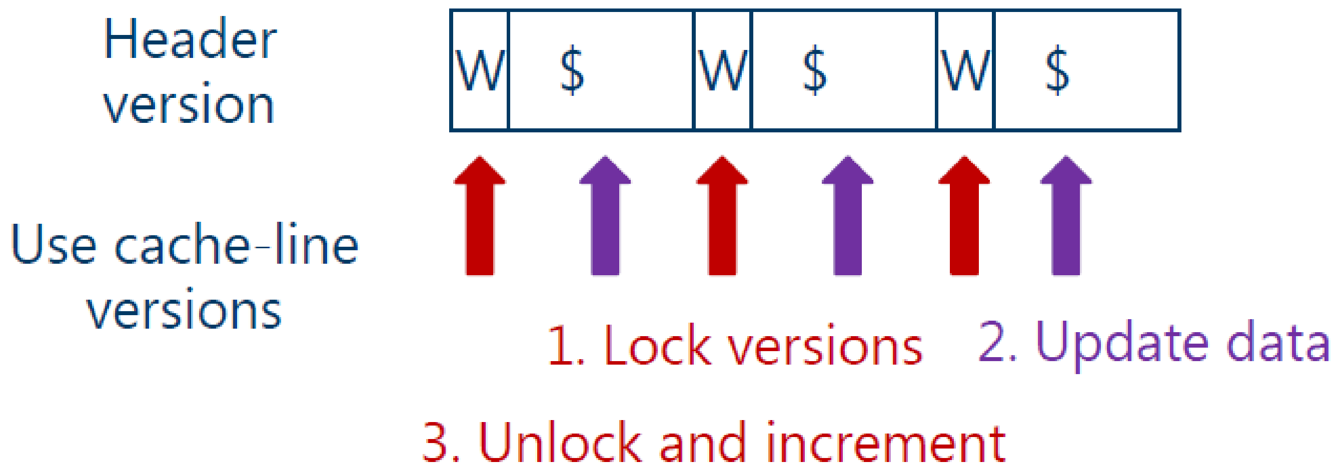
Traditional Lock-free reads

- Reading requires three accesses



FaRM Lock-free Reads

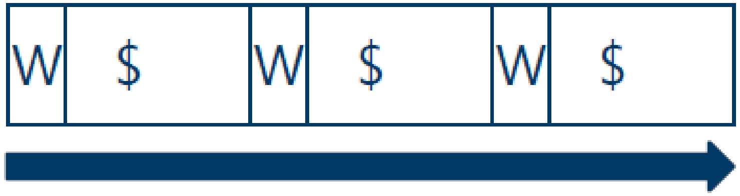
- FaRM relies on cache coherent DMA
- Version info in each of cache-lines



FaRM Lock-free Reads

- single RDMA read

Header
version

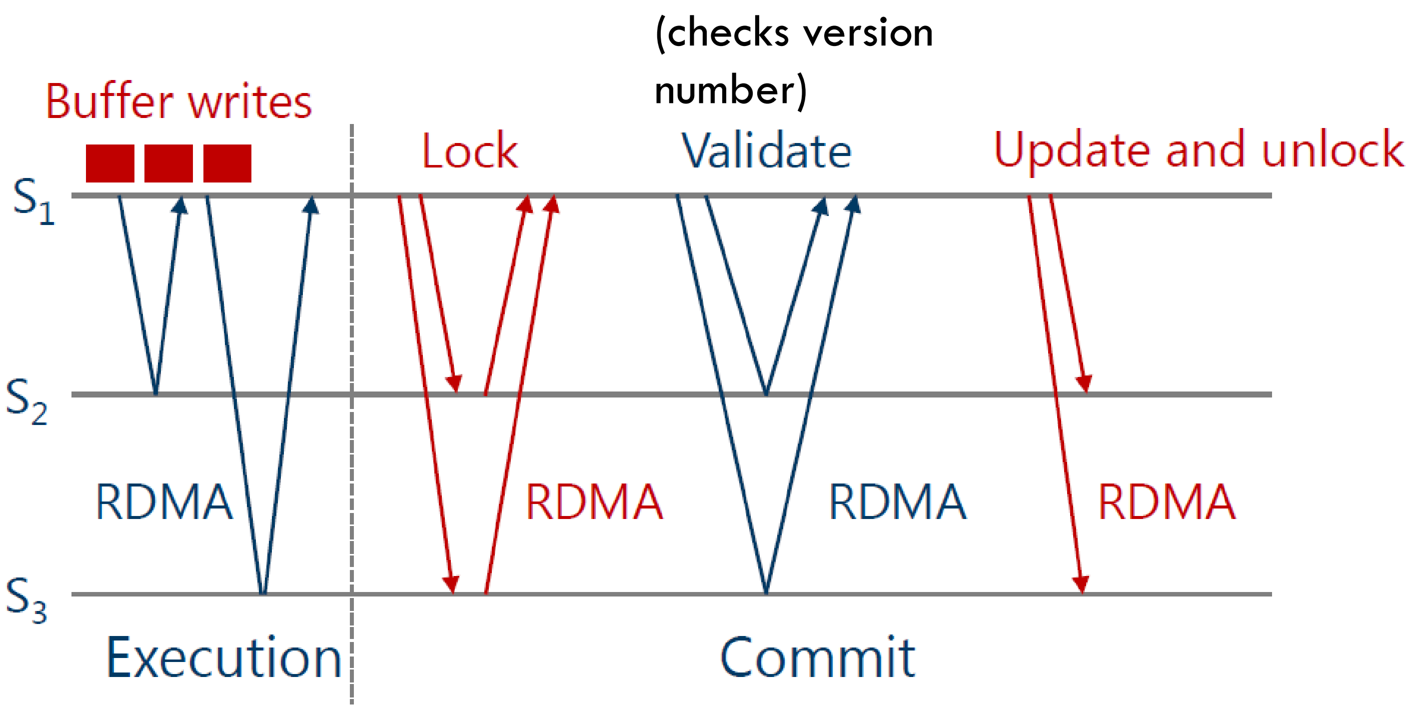


Cache-line
versions

One RDMA read of the whole object,
check that all versions are equal

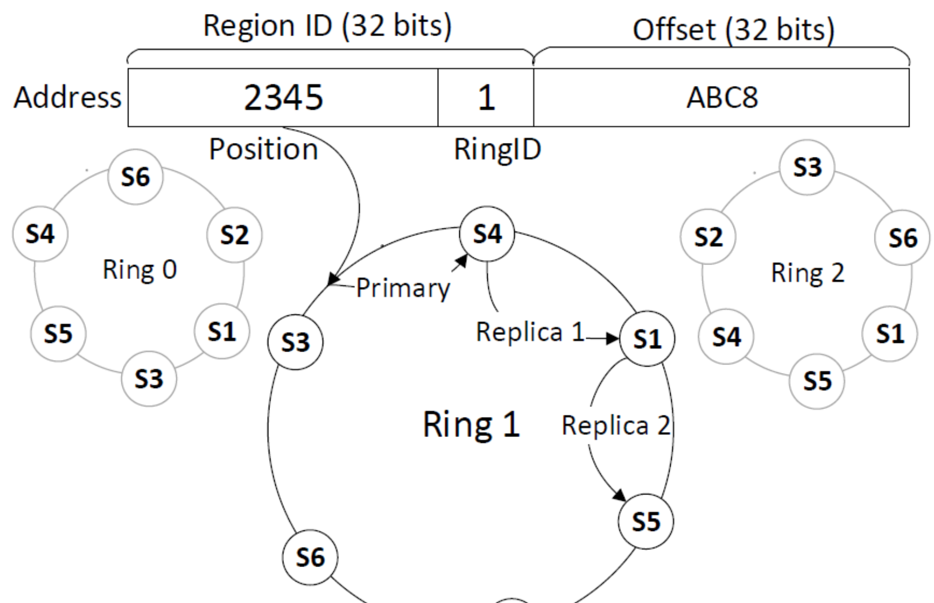
FaRM: Distributed Transactions

- general mechanism to ensure consistency
- Two-stage commits



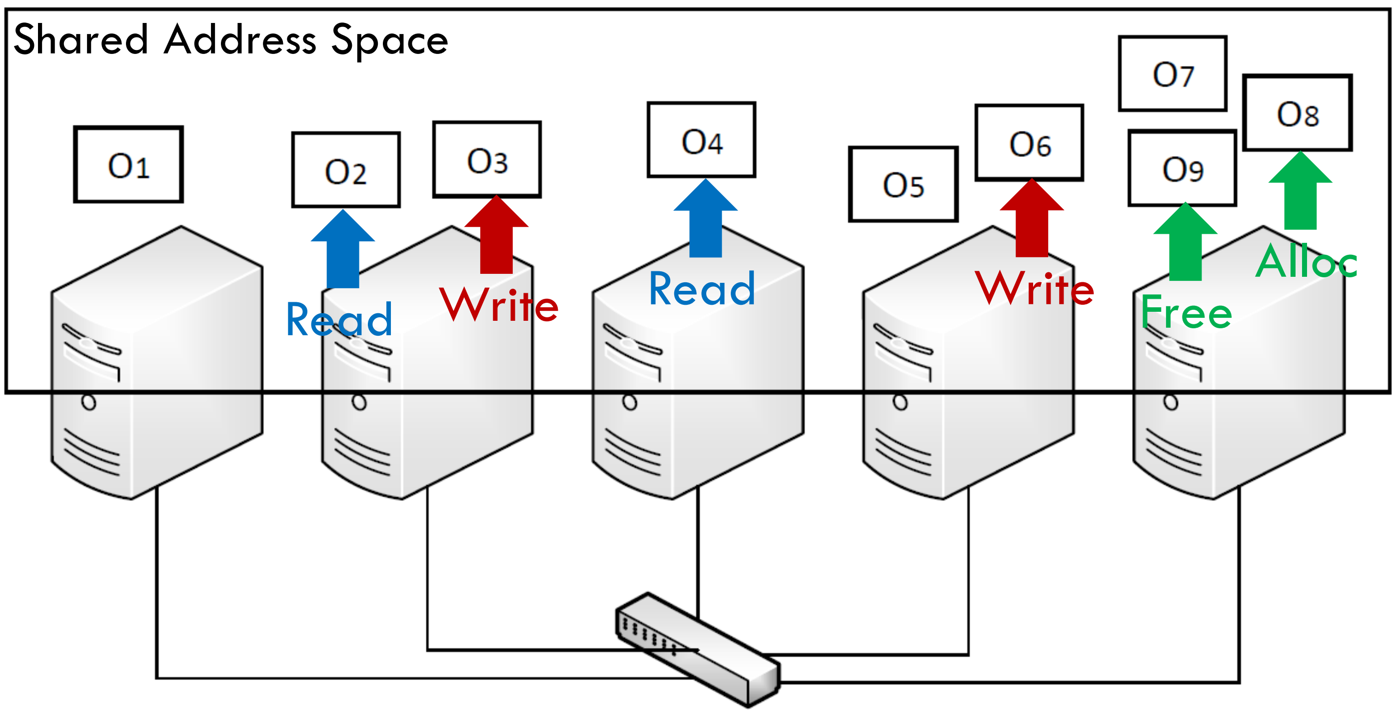
Shared Address Space

- shared address space consists of many shared memory regions
- consistent hashing for mapping region identifier to the machine that stores the object
  - each machine is mapped into k virtual rings



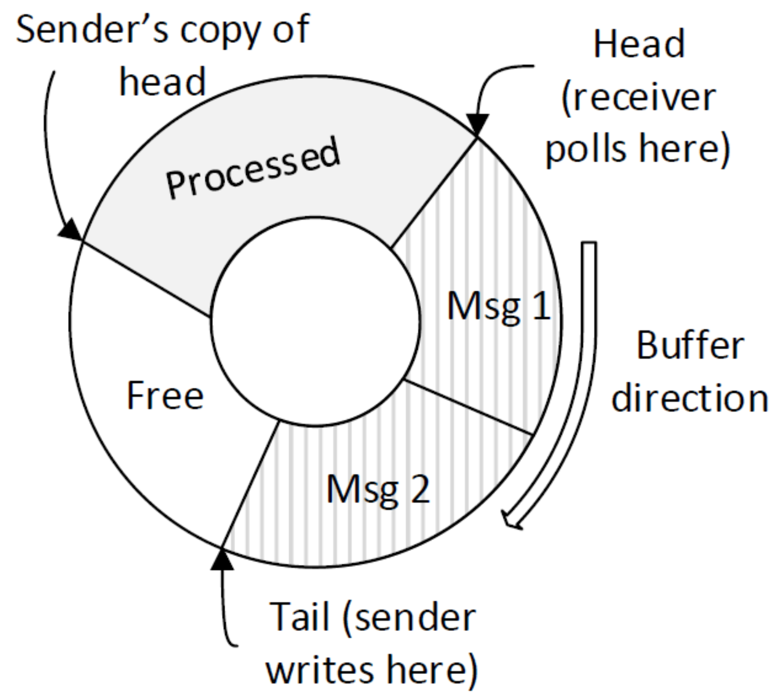
Transactions in Shared Address Space

- Strong consistency
- Atomic execution of multiple operations



Communication Primitives

- One-sided RDMA reads
  - to access data directly
- RDMA writes
  - circular buffer is used for unidirectional channel
  - one buffer for each sender/receiver pair
  - buffer is stored on receiver



benchmark on communication primitives

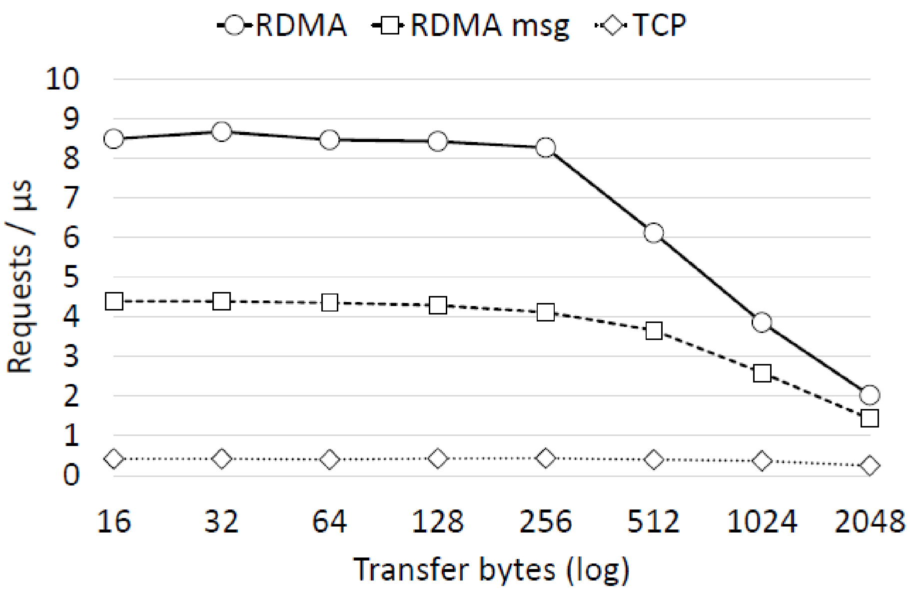


Figure 2: Random reads: request rate per machine

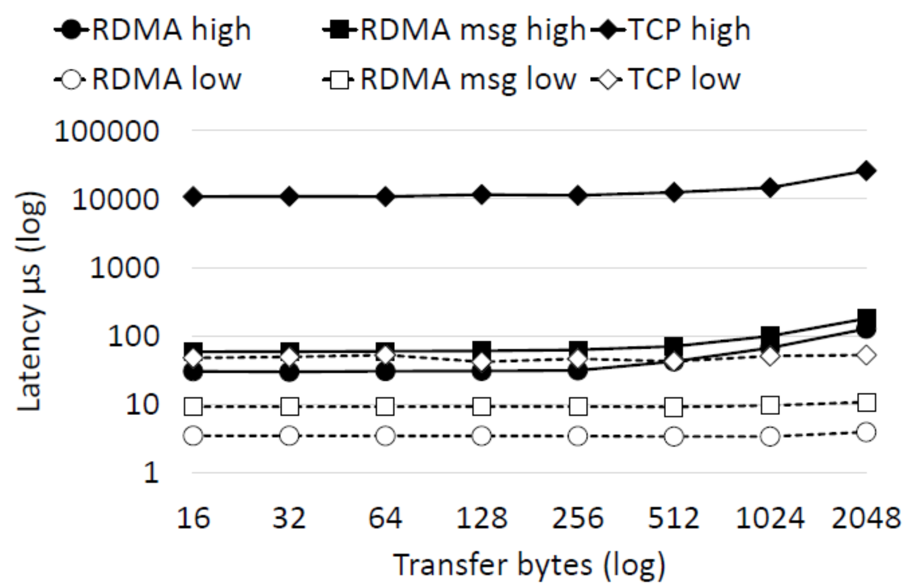
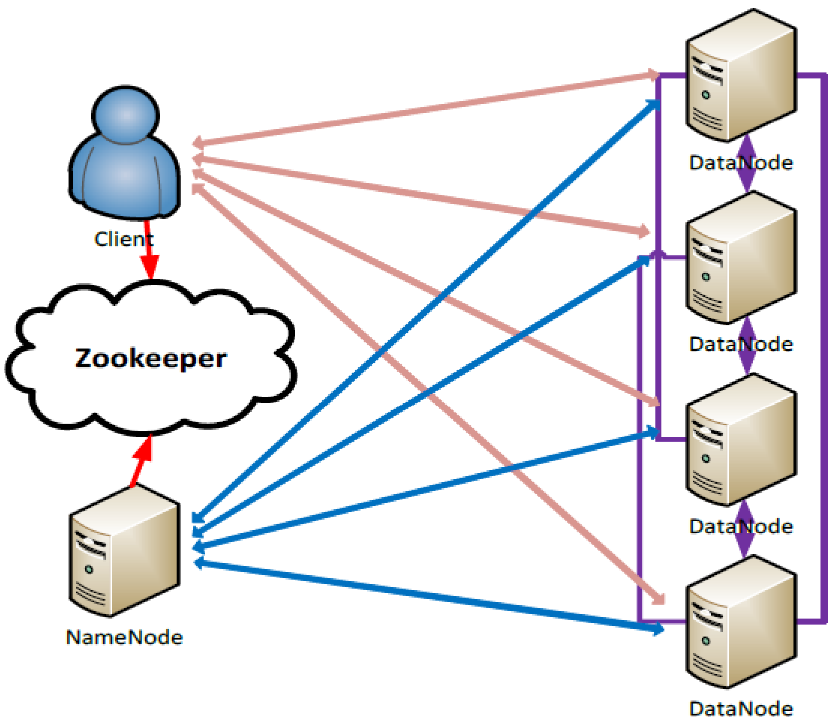


Figure 3: Random reads: latency with high and low load

Limited cache space in NIC

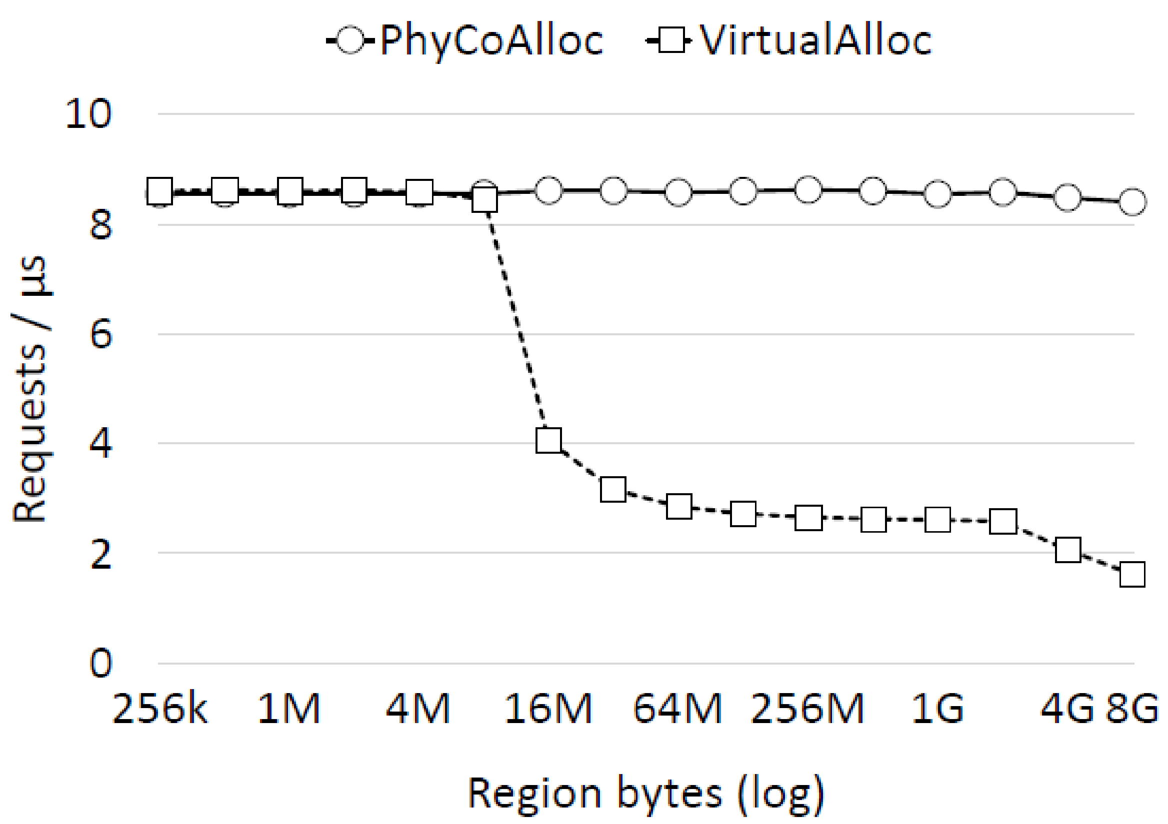
- Some Hadoop clusters can have hundreds and thousands of nodes
- Performance of RDMA can suffer as amount of memory registered increases
  - NIC will run out of space to cache all page tables



Limited cache space in NIC

- FaRM's solution: PhyCo
  - kernel driver that allocates a large number of physically contiguous and naturally aligned 2GB memory regions at boot time
  - maps the region into the virtual address space aligned on a 2GB boundary

PhyCo

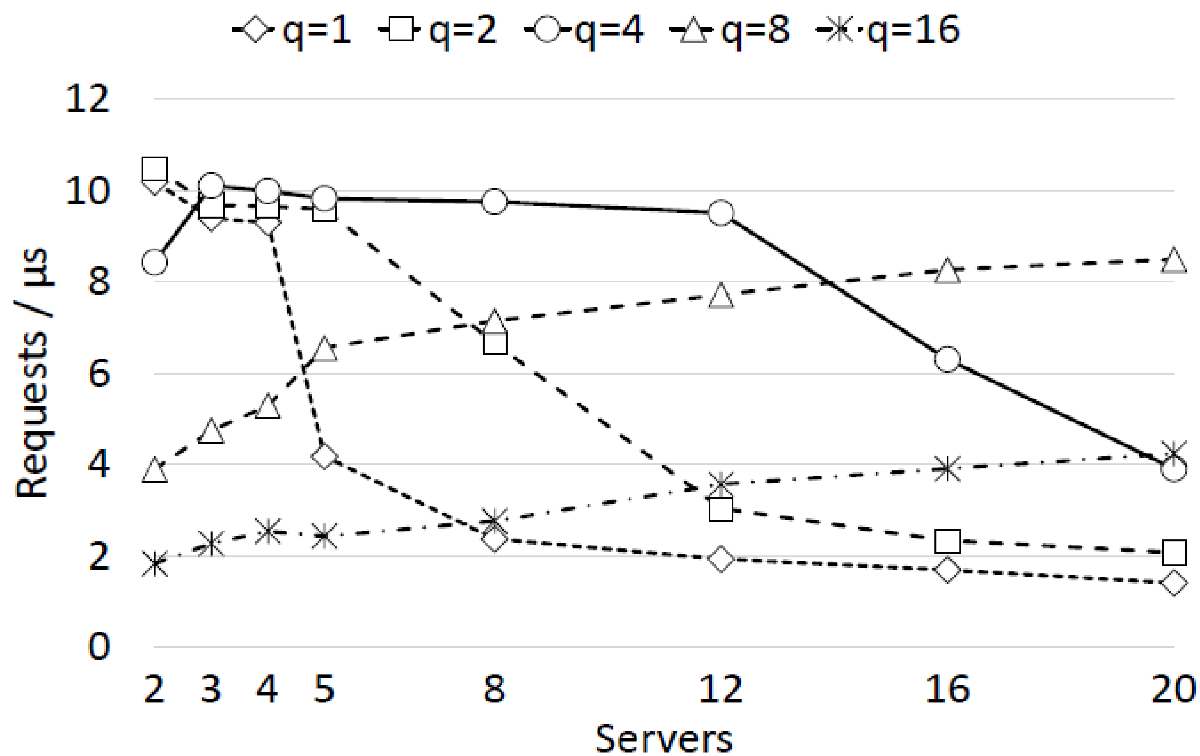

PhyCoAlloc
VirtualAlloc
10
8
6
4
2
0
Requests / µs
256k
1M
4M
16M
64M
256M
1G
4G
8G
Region bytes (log)

Limited cache space in NIC

- PhyCo still suffered as number of clusters increased
  - because it can run out of space to cache all queue pair
    - $2 \times m \times t^2$ queue pairs per machine
      - $m =$ number of machines, $t =$ number of threads per machine
  - single connection between a thread and each remote machine
    - $2 \times m \times t$
  - queue pair sharing among q threads
    - $2 \times m \times t / q$

Connection Multiplexing

- best value of q depends on cluster size



Experiments

- Key-value store: lookup scalability

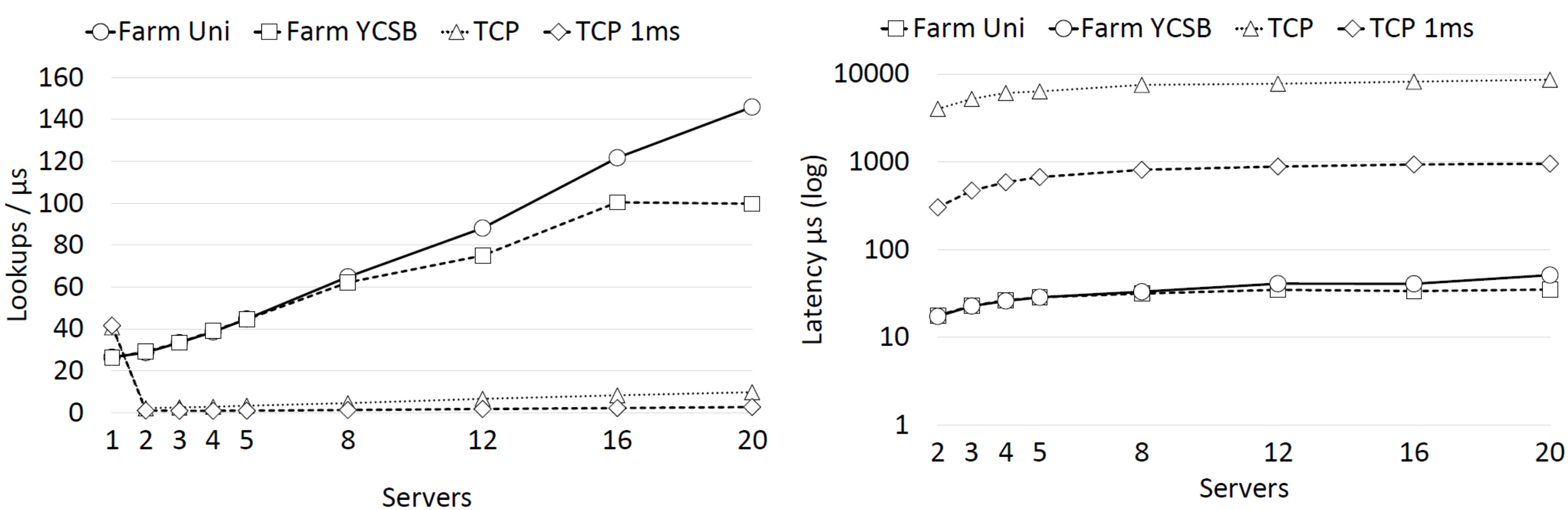
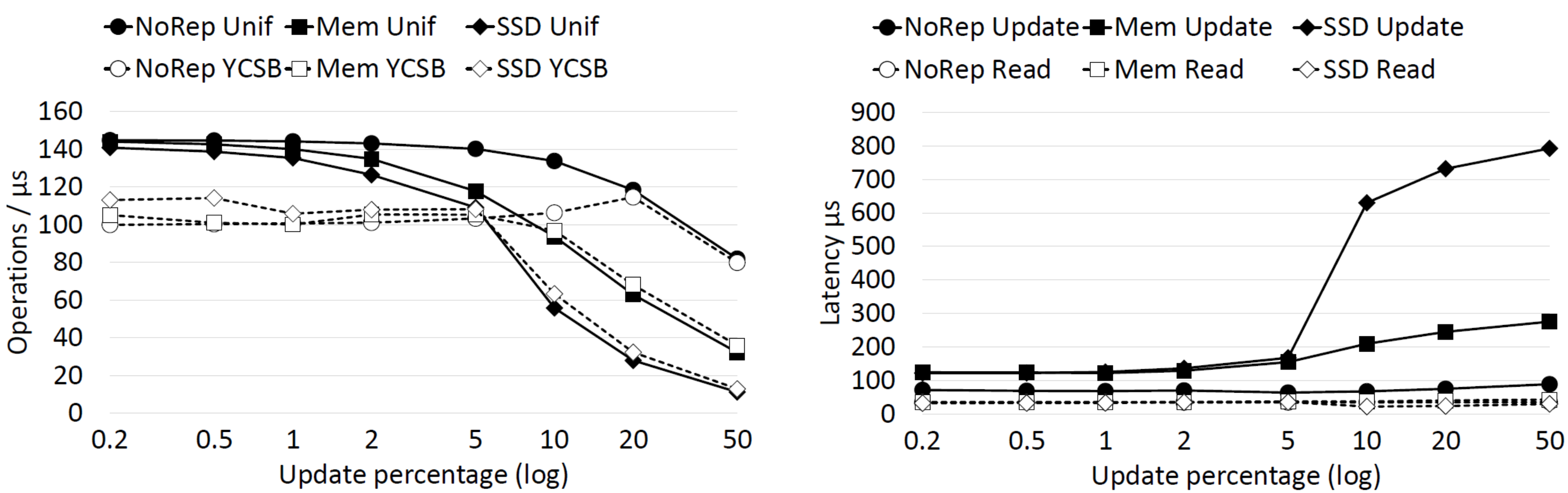


Figure 12: Key-value store: lookup scalability

Experiments

- Key-value store: varying update rates



(a) Throughput (YCSB and uniform)

(b) Latency for lookups and updates (uniform)

Figure 15: Key-value store: varying update rates