C-Hint: An Effective and Reliable Cache Management for RDMA-Accelerated Key-Value Stores

Yandong Wang, Xiaoqiao Meng, Li Zhang, Jian Tan

Presented by: Guangxiang Du

What’s RDMA and why RDMA?

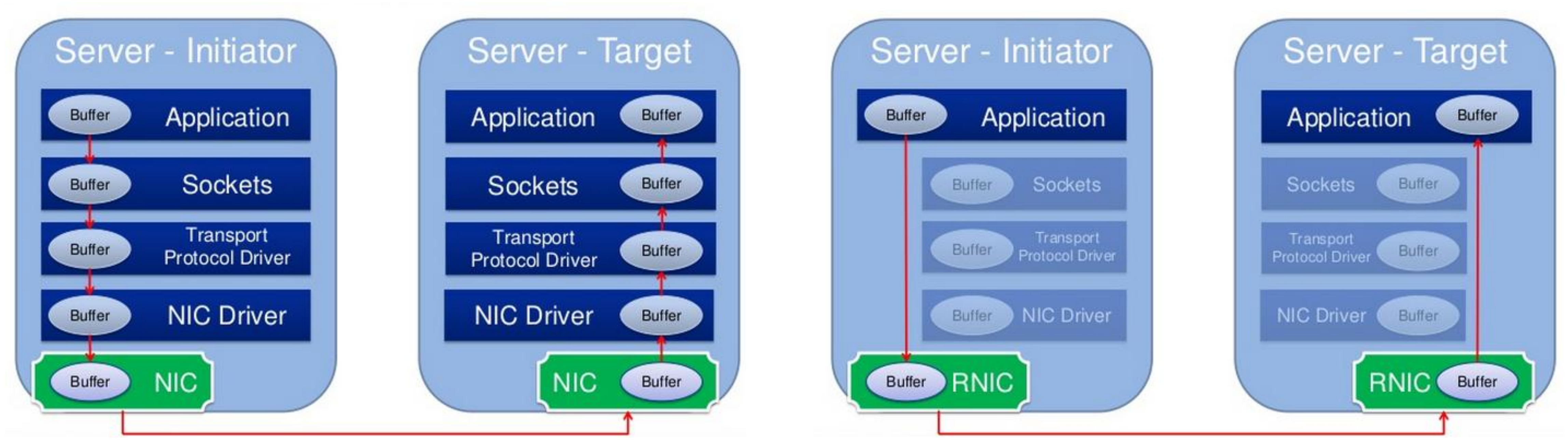
Definition:
RDMA is the ability of accessing memory on a remote machine without interrupting the processing of the CPU on remote machine.

Characteristics:

-Kernel Bypass: Applications can perform data transfer directly from userspace, without the need of data-copy and processing in network software layer (TCP/IP) in OS kernel.

-No remote CPU involvement: Applications can access remote memory without consuming any CPU power in the remote machine.

cont'd



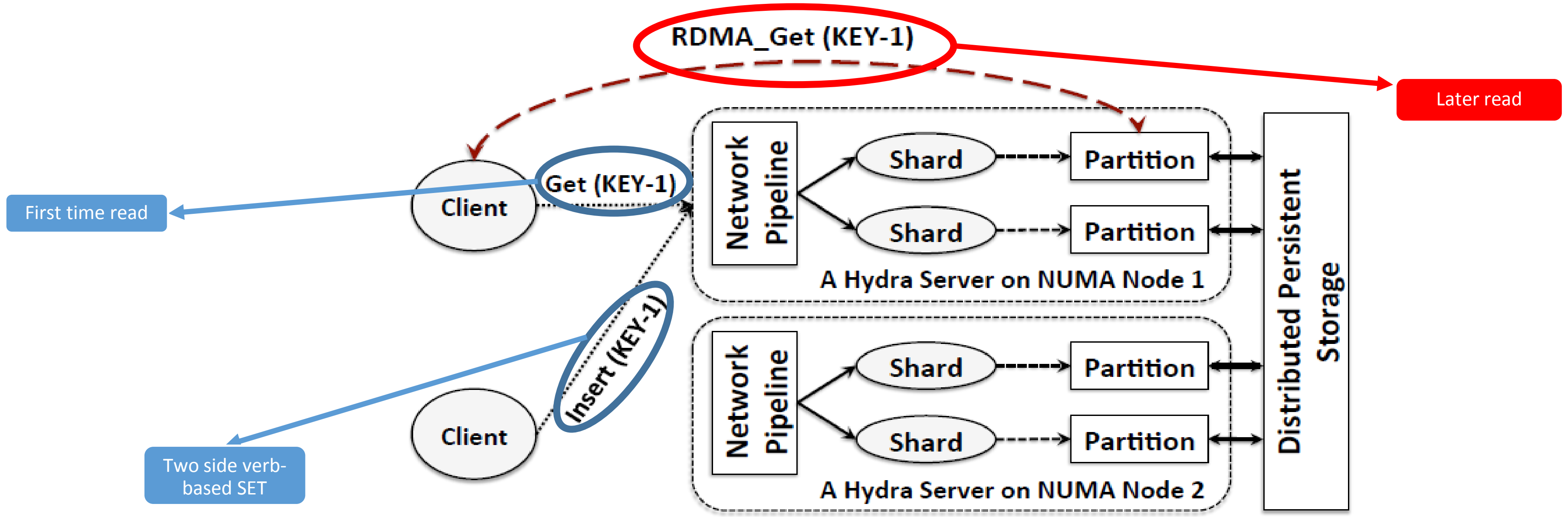
Benefits:

-Low latency

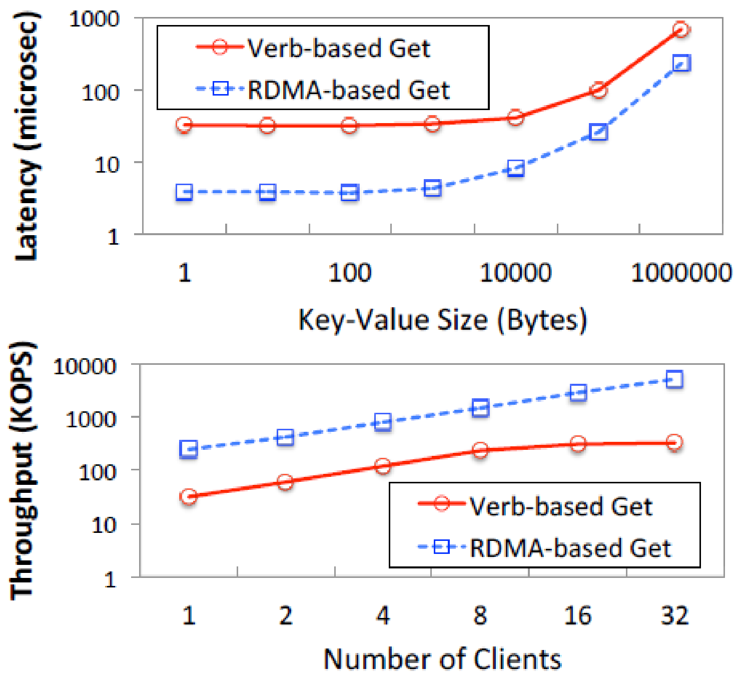
-High Throughput

-Low Remote side CPU footprint

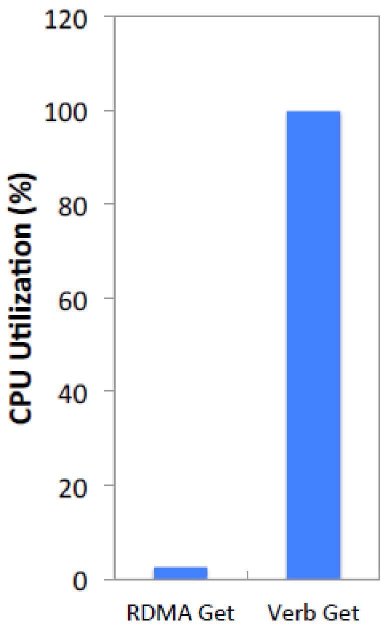
System Background



- Clients have operations like GET, SET, REMOVE, etc.
- Using two-side Verb-based Send/Recv for SET, REMOVE
- First time access to an item with two-side Send/Recv (obtain remote address), later GET operations will be one-side RDMA read



(a) Latency and Throughput Comparison



(b) CPU Utilization Comparison

- one side RDMA read and server write create race condition?
- There have been several solutions proposed, such as checksum or cache line versioning.

Challenges for Cache Management brought by RDMA

- Server Unaware of RDMA reads, difficult to keep track of popularity of cached items. (inefficient cache replacement scheme leads to severe performance downgrade)

- when the server evicts an items, it needs to invalidate the remote pointer cached in the client side. (broadcast on every eviction is significant overhead)

Overview of C-Hint

- Goals:
- 1) deliver sustainable high hit rate with limited cache capacity.
- 2) not to compromise the benefit of RDMA read
- 3) provide reliable resource reclamation scheme.

- Design decisions:
- 1) Client-Assisted Cache Management
- 2) Managing Key-Value Pairs via Leases
- 3) Popularity Differentiated Lease Assignment
- 4) Classifying Hot-Cold Items for Access History Report
- 5) Delayed Memory Reclamation

Client-Assisted Cache Management

- As said before, server alone can’t track the accesses of the cached items.

- Therefore, Clients are required to propagate partial access history of key-value pairs to server periodically.
- The server aggregate the information to establish a global view of access pattern.

Managing Key-Value Pairs via Leases

- The remote pointer contains a time range (lease) within which the client is allowed to conduct GET with RDMA read. (server side guarantee that the particular item is available and not corrupted until the lease expires.)

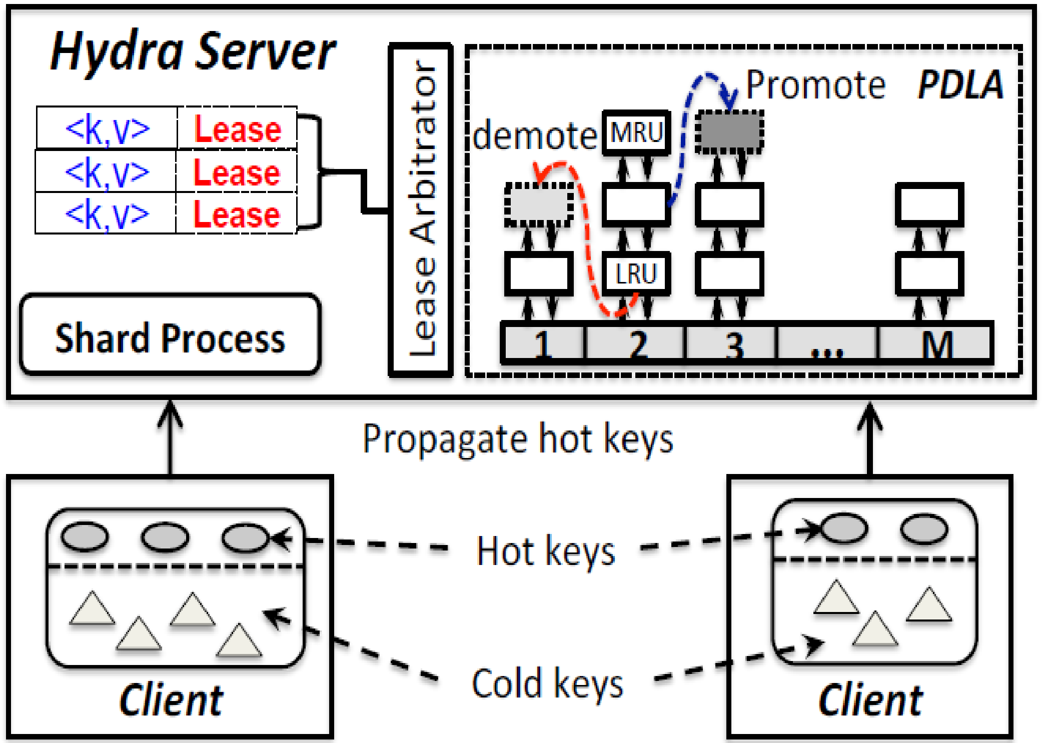
- Lease will not be extended unless the clients send a verb-based renewing message to server asking for extension.

Popularity Differentiated Lease Assignment

- How long a lease term should be?
- too short? Remote pointers cached by clients get invalidated frequently, low utilization of RDMA.
- too long? Unpopular items consume much of the space of server cache.
- Static determined lease term? Or changing dynamically?

- C-hint use approximation of Multi-queue algorithm to determine the lease term for different key-value pairs adaptively according to their recency and popularity.

Cont’d



-Multiple LRU queues

-qid = log2(c), c is approximate access count, updated on server-aware operations

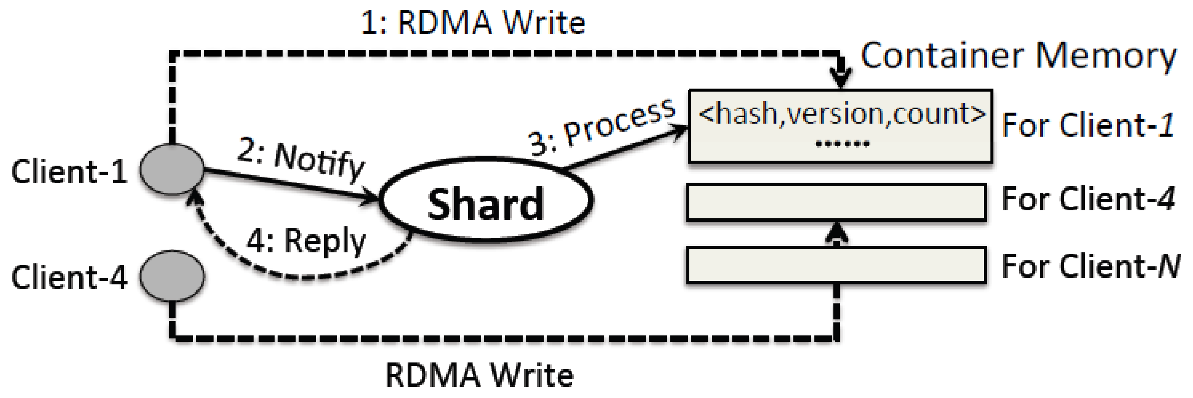
- term = qid * α

- If head items of LRU queues expires, demoted to queues of smaller id (take into account of aging popularity)

- Intuition: Try to assign recent, popular item with long lease term.

Classifying Hot-Cold Items for Access History Report

- Server needs to aggregate history report from lots of clients, heavy loading, harmful to QoS of latency sensitive operations.
- Clients use Adaptive Replacement Algorithm (2 LRU queues) to only report history of hot items. (smaller report)
- Further optimization: clients use RDMA write to write reports to dedicated memory locations.



Delayed Memory Reclamation

- After REMOVE, C-hint mark item deleted, but reclaiming the memory only after the lease expires.

- Avoid expensive broadcast to invalidate the client-cached remote pointers.

Performance Evaluation

- Hit Ratio

- Comparison between:
  - Original HydraDB (random replacement scheme)
  - Faithful LRU (disabled RDMA, as baseline)
  - Apprxi-LRU (C-hint design, except not Multi-queue, single LRU queue)
  - Faithful Multi-queue (disabled RDMA, as baseline)
  - C-Hint design

Experimental Environment

- 5 x86_64 machines connected through InfiniBand
  - 4 server instances on 2 different machines, clients on the rest

- Benchmark
  - Yahoo! Cloud Serving Benchmark (YCSB 0.14)
  - Generate 3 workloads, (100% GET + 0% SET) (95% GET + 5% SET) (90% GET + 10% SET)
  - Each 300 million operation, 80 million key-value pairs (23B key + 1KB value)

Hit Ratio Analysis

- $Total\ Hit\ Ratio = \frac{RDMA\_Get\ Hit + Verb\_Get\ Hit}{Total\ Get\ Operations}$
- $Remote\ Pointer\ Hit\ Ratio = \frac{RDMA\_Get\ Hit}{Total\ Get\ Operations}$

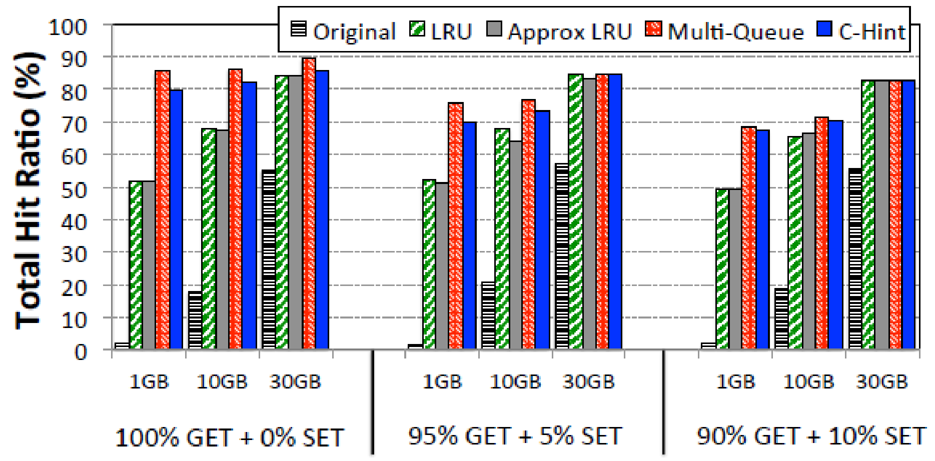


Figure 5: Overall hit ratio comparison between C-Hint and the other four solutions.

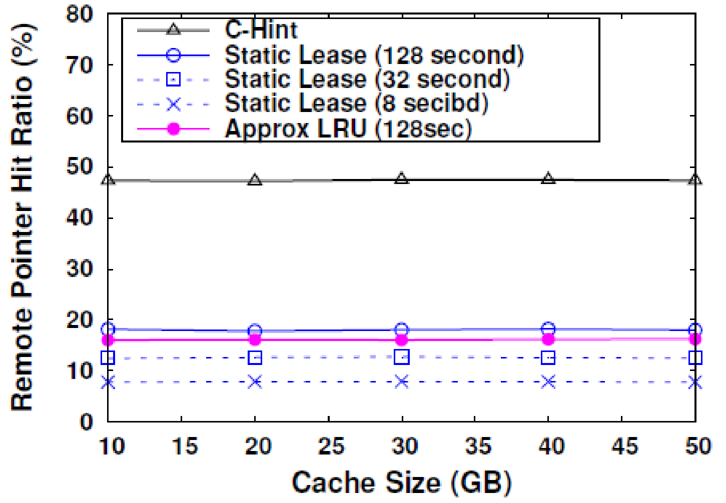
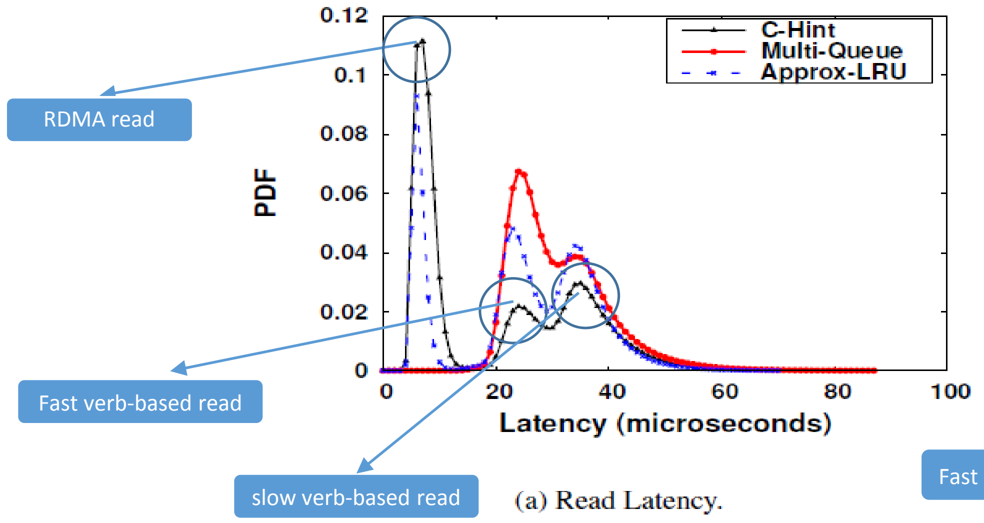
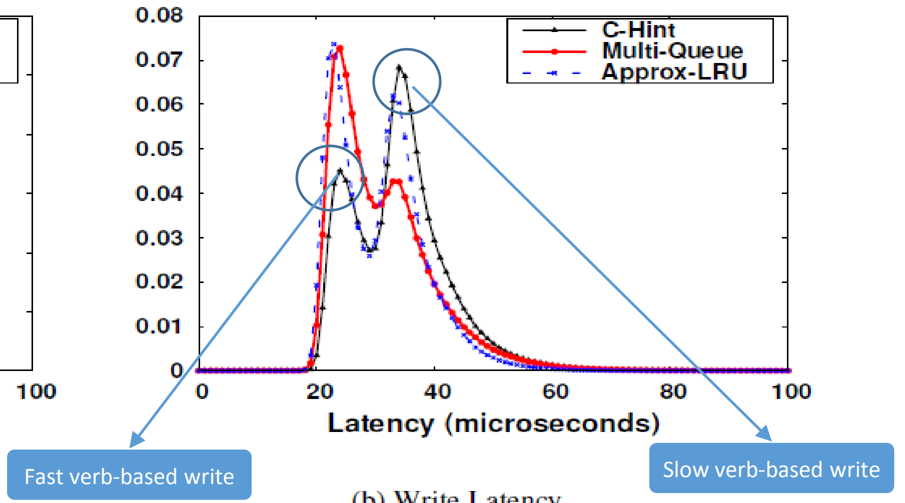


Figure 6: Hit Ratio of remote pointers cached on the client side with 95%GET + 5%SET workload.

Latency Analysis

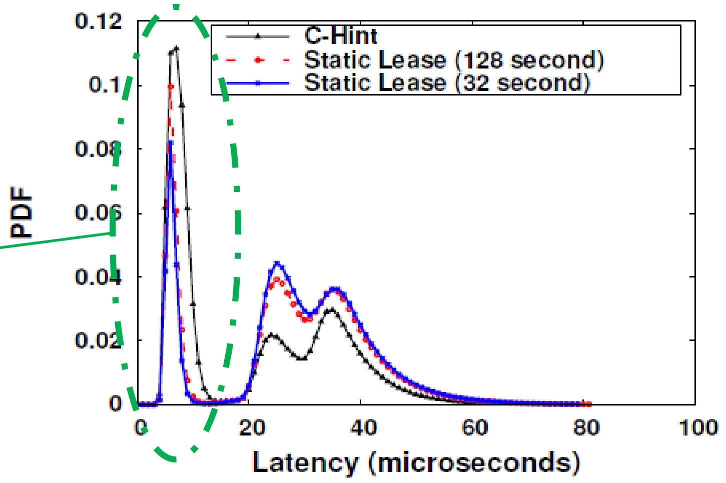


(a) Read Latency.



(b) Write Latency.

Figure 7: Latency comparison between C-Hint, Multi-Queue, and Approx-LRU as well by using 95%GET + 5%SET workload.



There are more requests whose latencies are below 20 us in the c-hint curve, which respond to rdma read.

That means PDLA is better that static determined lease term in terms of rdma utilization

Figure 8: C-Hint can efficiently improve the GET latency by avoiding statically assigning the lease terms without differentiating distinct access patterns.

Impact of Access History Report

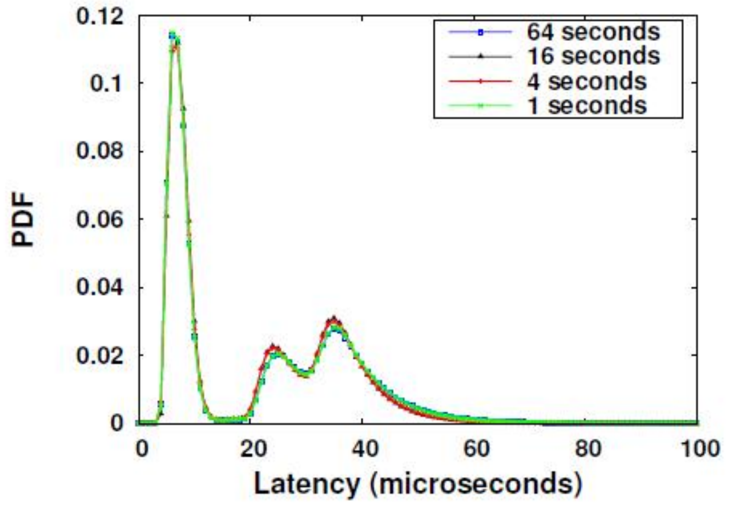


Figure 9: Frequently reporting the access history imposes negligible performance overhead.

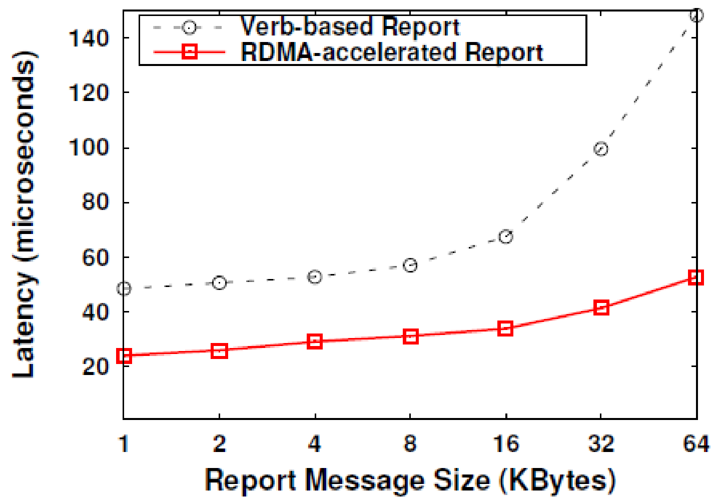


Figure 10: Using RDMA Write to conduct history report can efficiently save communication cost.

Cons/Criticism/thoughts

- 1) vulnerable to malicious clients in a wild environment
  - Defer to future research topic
- 2) the scale of experiment may not be enough
  - Only with 5 machines and 4 servers instances
- 3) unsynchronized clock between clients and server lead to inconsistent view of lease term

Summary

- C-Hint addresses 3 challenges:
  - 1) how to track popularity of cached items although server unaware of RDMA reads
  - 2) effectively, reliably reclaim memory on server
  - 3) improve the hit ratio without compromising the performance benefit of RDMA