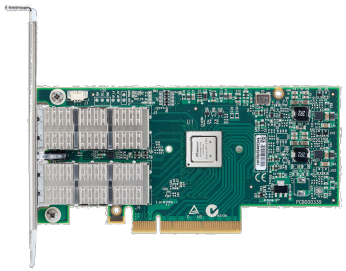
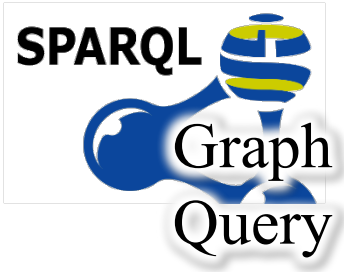




Improving Large-scale Systems with RDMA and HTM

A Perspective from Systems Software

RONG CHEN

http://ipads.se.sjtu.edu.cn/rong_chen

Institute of Parallel and Distributed Systems
Shanghai Jiao Tong University

Joint work with Haibo, Xinda, Jiaxin, Yanzhe, Youyang@SJTU, and the Wukong work is also with Fefei@Utah

Business Demand – High Throughput

Peak transactions per second:

175,000 new orders

120,000 payment

Business Demand – Low Latency

Evolution and Practice: Low-latency Distributed Applications in Finance

The finance industry has unique demands for low-latency distributed systems

- Read input message from network and parse – 5 microseconds
- Look up client profile – 3.2 milliseconds (3,200 microseconds)
- Compute client quote – 15 microseconds
- Log quote – 20 microseconds
- Serialize quote to a response message – 5 microseconds
- Write to network – 5 microseconds

How to Do It? Conventional Approach

TPC-C World Record

504,161 TXs/second
Cost: 30,528,863 USD

172,770 TXs/second
Cost: 14,276,808 USD

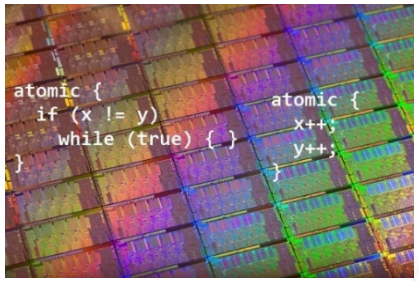
http://www.tpc.org/tpcc/results/tpcc_results.asp?print=false&orderby=tpm&sortby=desc

How to Do It? Today's Approach

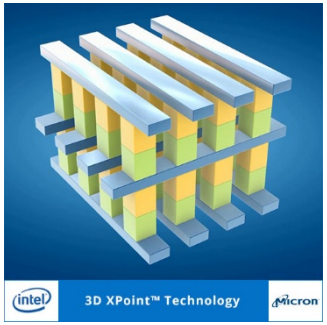
OceanBase **MySQL Cluster**

Hardware Platform

HTM

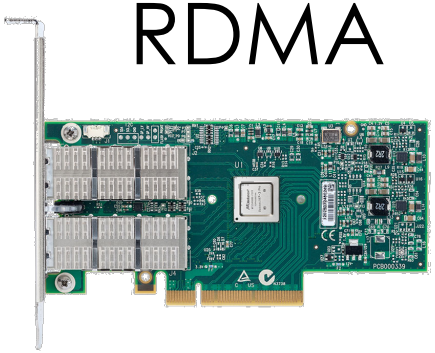


NVM



GPU

RDMA



A few rack-scale machines

RDMA: Remote Direct Memory Access

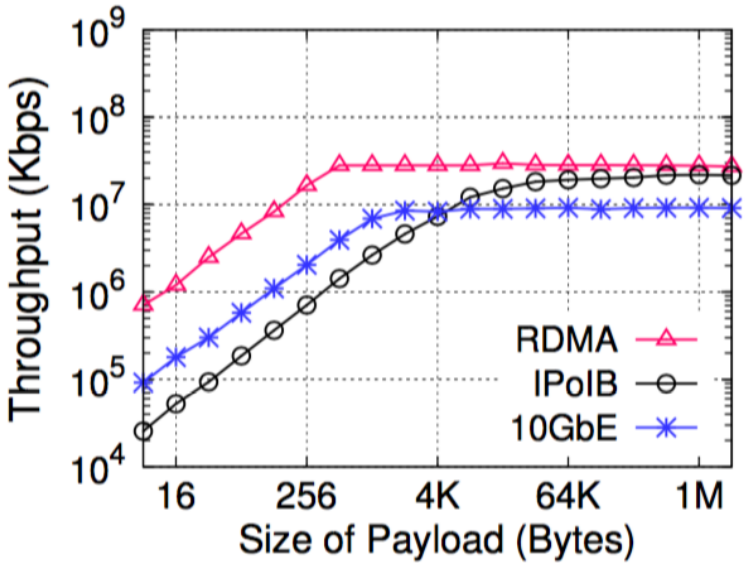
A network feature that allows **direct** access to the memory of a **remote** computer

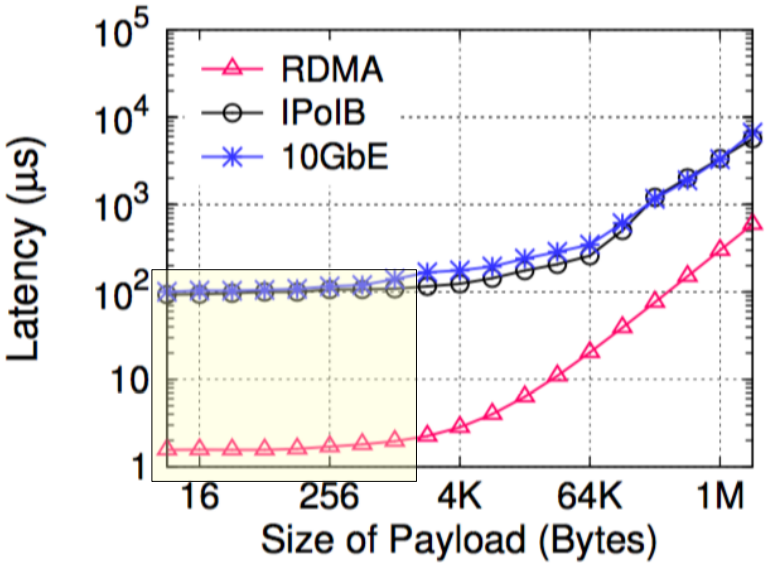
High speed, low latency & low CPU overhead

- Interface: IPoIB, two-sided RDMA (SEND/RECV), and one-sided RDMA (READ/WRITE/CAS)
- Bypasses OS kernels: Zero copy
  Bypasses CPU (one-sided)
- Round-trip time: one-sided/~3μs,
  two-sided/~7μs, IPoIB/~100 μs

One-sided RDMA Performance

Performance of Random Read[1]





[1] Mellanox ConnectX-3 MCX353A 56Gbps InfiniBand NIC

In-memory KV-Stores

Interface: **GET**, **PUT**, etc.

A key pillar for many systems

- Data cache (e.g., Memcached in FB)
- In-memory database (e.g., OLTP, OLAP)
- Graph query processing

Key requirements

- Low latency
- High request rate (Throughput)

Key Questions

Communication mode btw. machines

- Symmetric vs. asymmetric

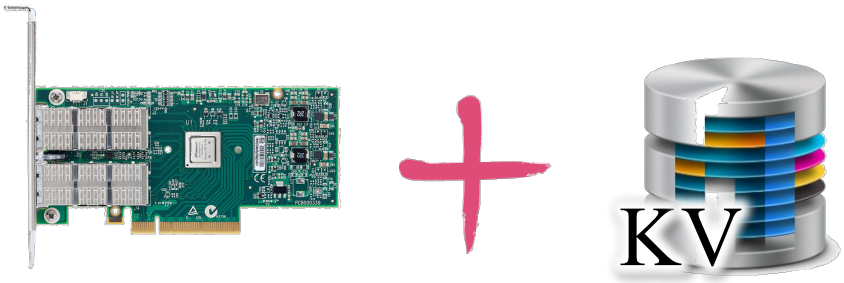
Data structure in KVS for RDMA

- Hash-based vs. tree-based
- How to store keys and values

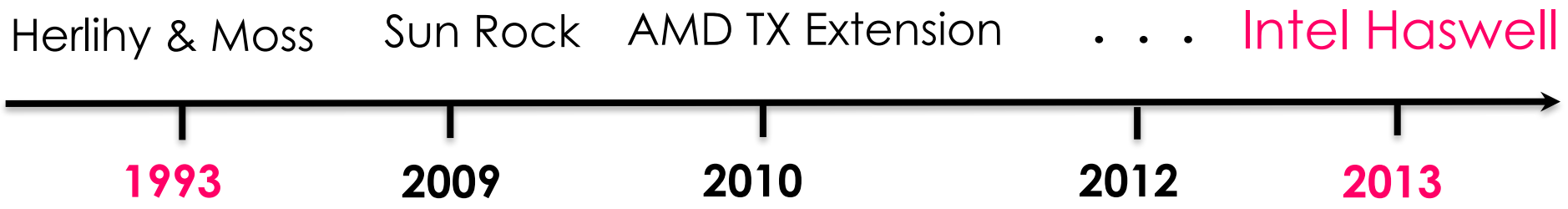
Interference between primitives

- Read vs. Write
- Local vs. Remote

RDMA-friendly Distributed Key-value Store



HTM: Hardware Transactional Memory



Massively available

Intel Haswell

Restricted Transactional Memory (RTM)

- Hardware transactional memory with limitations

Major limitations

- Working set is limited
- Some system events abort the TX

New instruction set

- Xbegin, Xend, Xabort

Programming with Intel RTM

RTM Usage

```
if _xbegin() == _XBEGIN_STARTED
        do some critical work
    _xend()
else
    fallback routine
```

Handle the abort event

Opportunities: (not so) New HW Features

HTM: **H**ardware **T**ransaction **M**emory

- Allow a group of load & store instructions to execute in an atomic, consistent and isolated (ACI) way

RDMA: **R**emote **D**irect **M**emory **A**ccess

- Provide cross-machine accesses with high speed, low latency and low CPU overhead

Rethink the design of **low-COST**

Opportunities with HTM & RDMA

HTM: **H**ardware **T**ransaction **M**emory

a non-transactional code will unconditionally abort a transaction when their accesses conflict

Strong Atomicity

RDMA: **R**emote **D**irect **M**emory **A**ccess

Opportunities with HTM & RDMA

HTM: **H**ardware **T**ransaction **M**emory

> *a non-transactional code will unconditionally abort a transaction when their accesses conflict*

Strong Atomicity

RDMA: **R**emote **D**irect **M**emory **A**ccess

> *one-sided RDMA operations are cache-coherent with local accesses*

Strong Consistency

Challenges & Goals

How to maximize **RDMA** potentials?

Use one-sided RDMA READs and WRITEs

Little or no host CPU involvement

Simple/Fast KV accesses (no complex encoding or versioning)

- Otherwise, host CPU has to do complex processing during accesses

Symmetric accesses

- No trading off client resource for server efficiency

Prior systems

Prior systems (e.g. Pilaf[ATC'13] and FaRM[NSDI'14])

- Only leverage one-sided RDMA to read, not write
- Complicated INSERT: hard to leverage HTM
- No RDMA-friendly caching mechanism

Content-based caching (e.g. replication) is hard to perform strong-consistent read and write locally, especially using RDMA

	Pilaf	FaRM
Hashing	Cuckoo	Hopscotch
Race Detection	Checksum	Versioning
Remote Read	One-sided RDMA	
Remote Write	Messaging	
Caching	No	

RDMA & HTM provides a new design space

Prior Systems

	Pilaf	FaRM	HERD	DrTM
Model	Asymmetric	Symmetric	Asymmetric	Symmetric
Hashing	Cuckoo	Hopscotch	Chaining	Chaining
Remote Read	One-sided RDMA		Messaging	One-sided RMDA
Remote Write	Messaging			
Race Detection	Checksum	Versioning	Exclusive Accesses	L: HTM D: Lock
Transaction	No	Yes	No	Yes
Caching	No	No	No	Yes

1. A bias towards RDMA-based remote operations
2. Only leverage one-sided RDMA to read
3. No efficient RDMA-friendly caching scheme
4. Complicated INSERT: hard to leverage HTM

DrTM-KV's Design

- ✓ Simple hash structure to fully leverage HTM
- ✓ Decouple race detection from memory store
  - Use HTM to detect races
  - Use one-sided RDMA ops for remote read & write
- ✓ Location-based and fully transparent cache

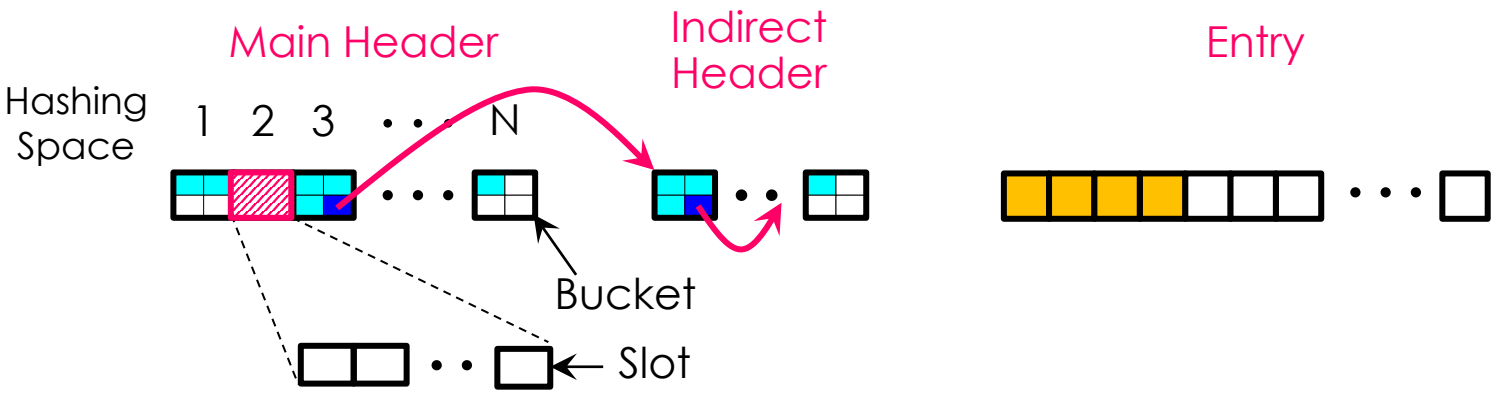
	Pilaf	FaRM	DrTM-kV
Hashing	Cuckoo	Hopscotch	Chaining
Race Detection	Checksum	Versioning	HTM
Remote Read	One-sided RDMA		One-sided RMDA
Remote Write	Messaging		
Caching	No		Yes

→ Simple

→ Efficient

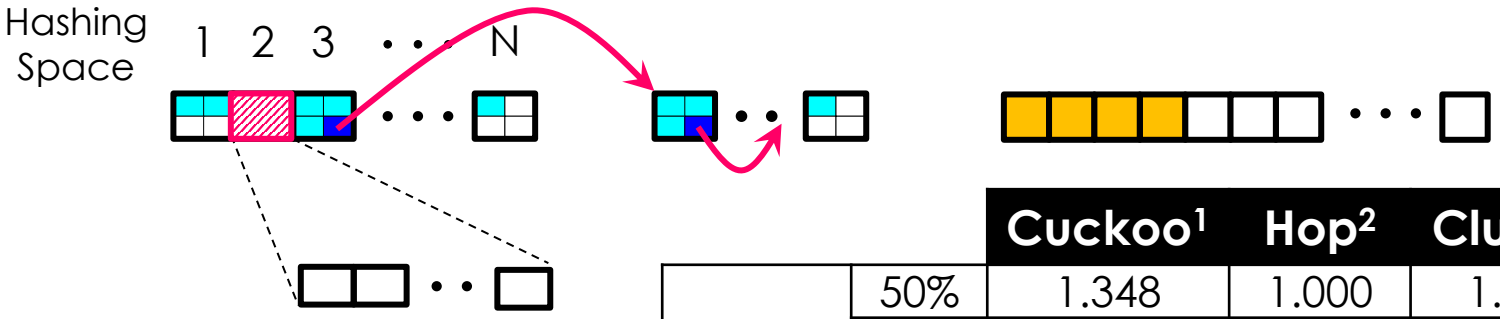
RDMA-friendly Cluster Chaining

- ✓ Similar to traditional chaining HT with associativity
- ✓ Decoupled memory region: index & data
- ✓ Shared indirect headers: high space efficiency



RDMA-friendly Cluster Chaining

- ✓ Similar to traditional chaining HT with associativity
- ✓ Decoupled memory region: index & data
- ✓ Shared indirect headers: high space efficiency



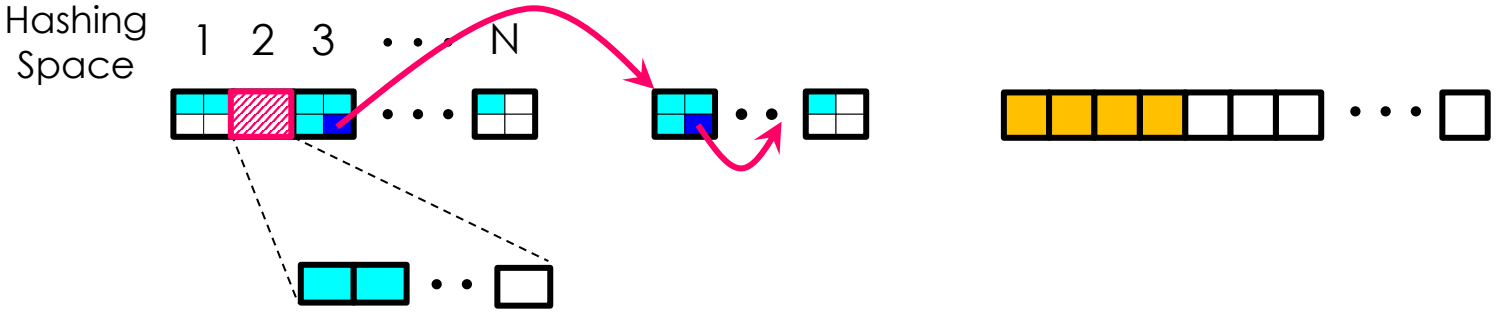
The average number of RDMA READs for lookups at different occupancies

		Cuckoo[1]	Hop[2]	Cluster[3]
Uniform	50%	1.348	1.000	1.008
	75%	1.652	1.011	1.052
	90%	1.956	1.044	1.100
Zipf θ=0.99	50%	1.304	1.000	1.004
	75%	1.712	1.020	1.039
	90%	1.924	1.040	1.091

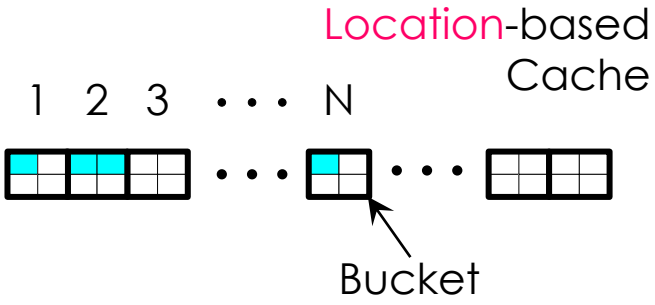
[1] Hopscotch hashing in FaRM configures the neighborhood with 8 (H=8).
[2] Cuckoo hashing in Pilaf uses 3 orthogonal hash functions and each bucket contains 1 slot.
[3] Cluster hashing in DrTM configures the associativity with 8.

Location-based Caching

- ✓ RDMA-friendly: focus on minimizing the lookup cost

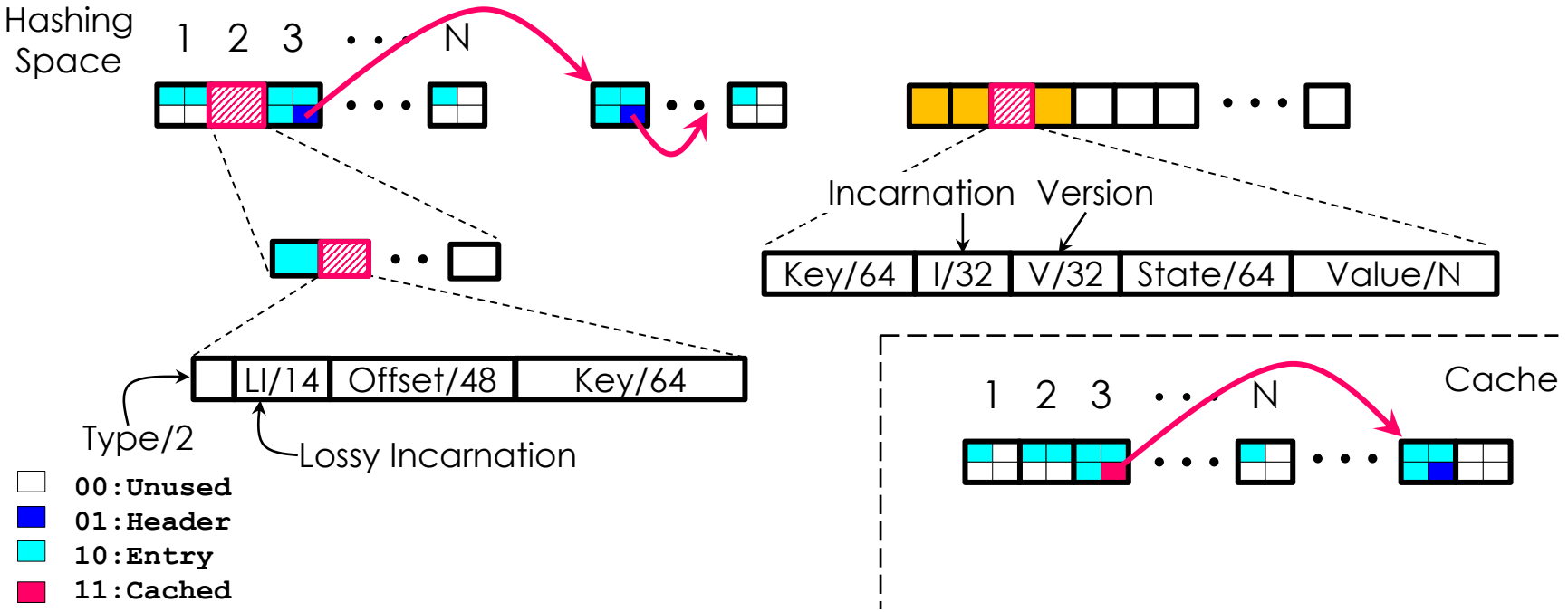


Treat cache as a partially stale snapshot of headers



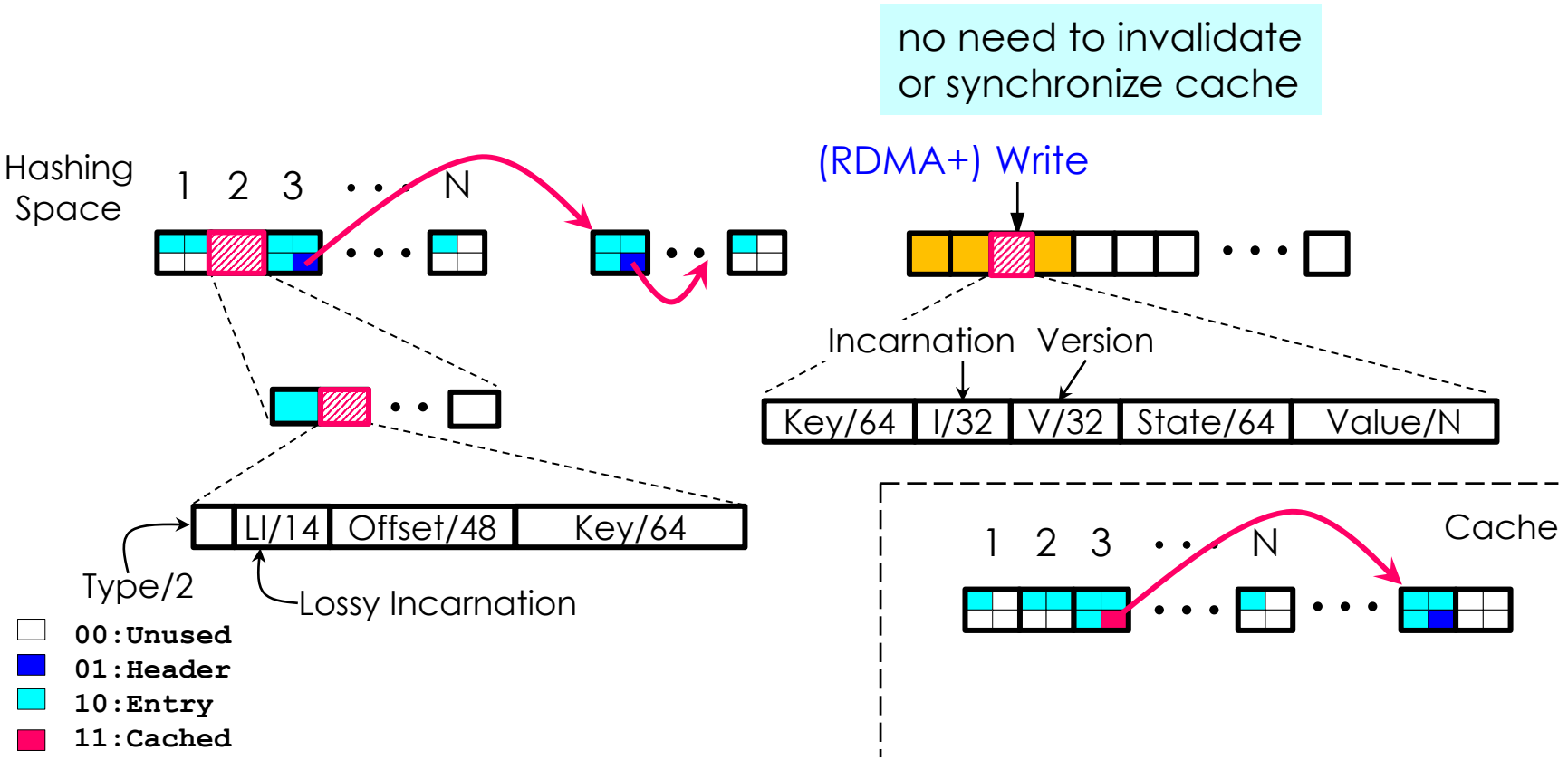
Location-based Caching

- ✓ RDMA-friendly: focus on minimizing the lookup cost
- ✓ Retain the full transparency to the host
  - All metadata used by other concurrency control mechanisms is encoded in the key-value entry



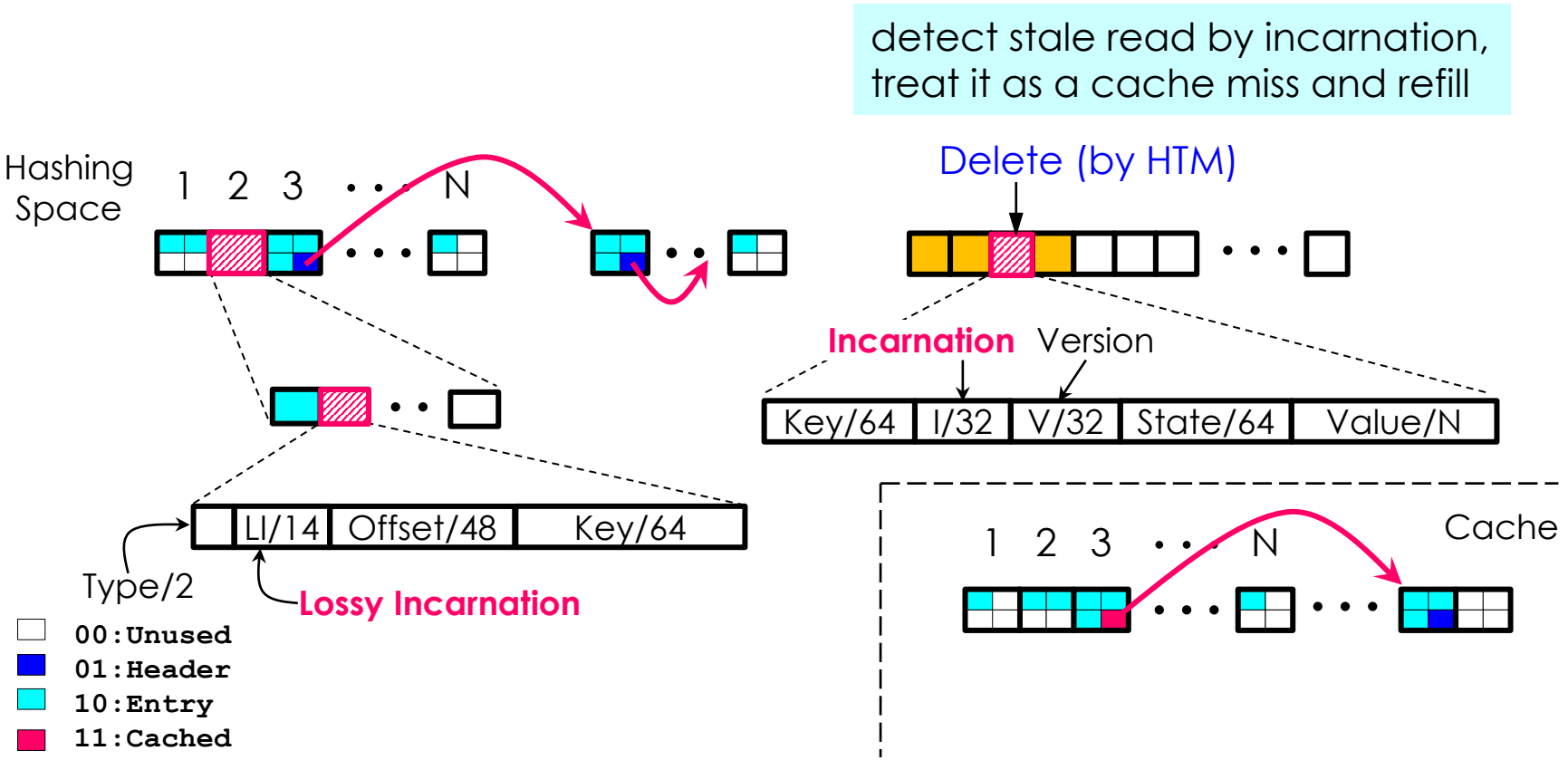
Location-based Caching

- ✓ RDMA-friendly: focus on minimizing the lookup cost
- ✓ Retain the full transparency to the host



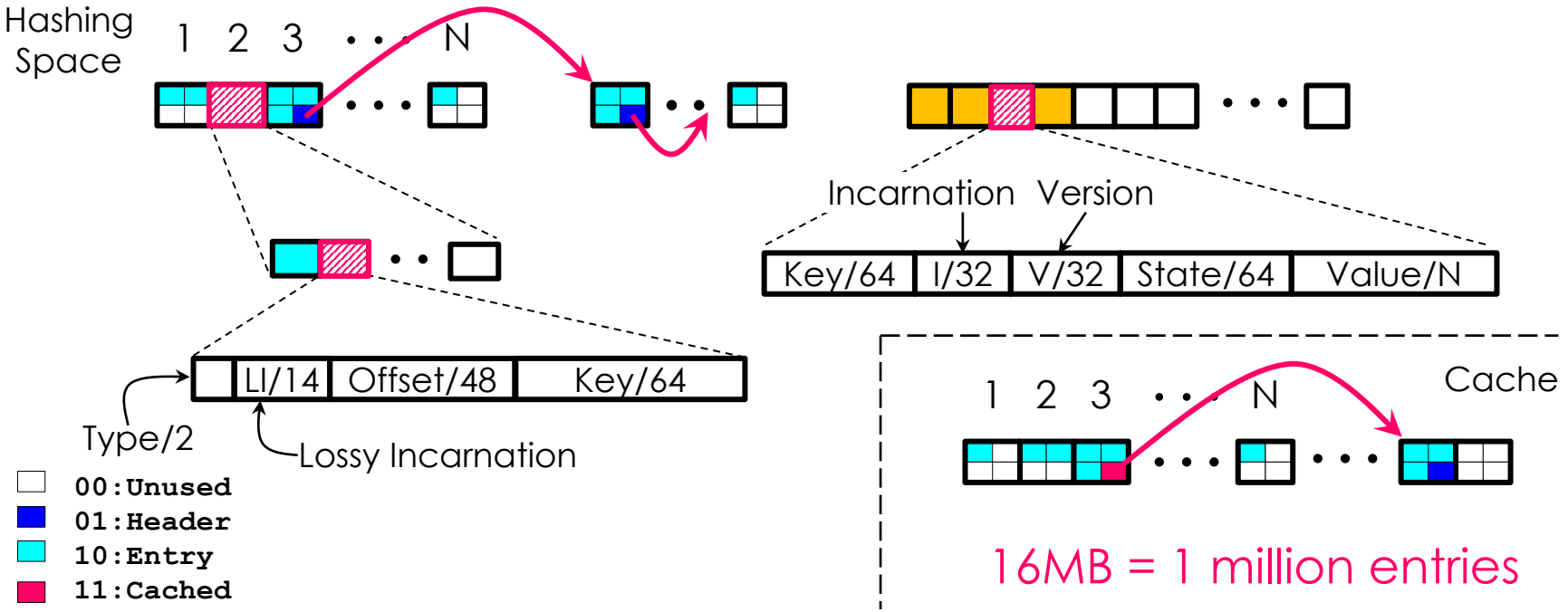
Location-based Caching

- ✓ RDMA-friendly: focus on minimizing the lookup cost
- ✓ Retain the full transparency to the host



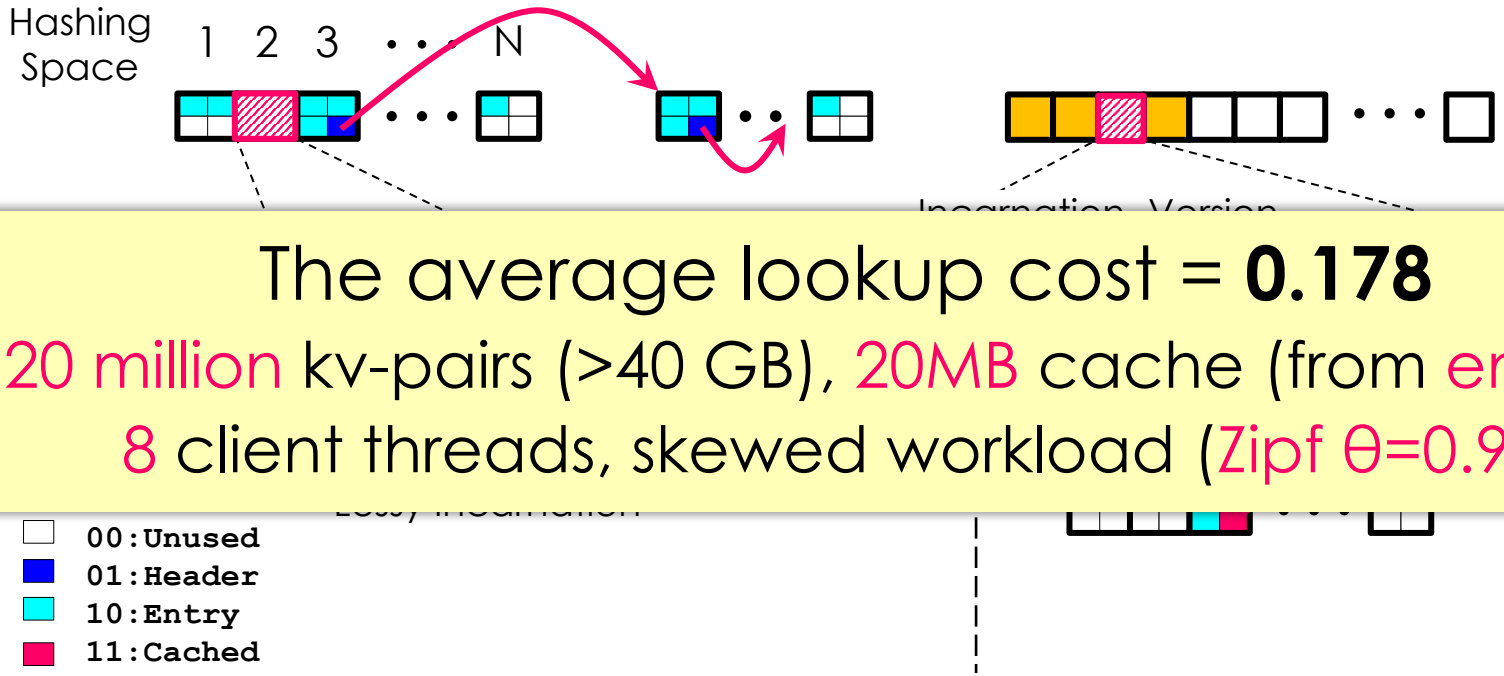
Location-based Caching

- ✓ RDMA-friendly: focus on minimizing the lookup cost
- ✓ Retain the full transparency to the host
- ✓ The size of cache for location is small



Location-based Caching

- ✓ RDMA-friendly: focus on minimizing the lookup cost
- ✓ Retain the full transparency to the host
- ✓ The size of cache for location is small
- ✓ All client threads can directly share the cache

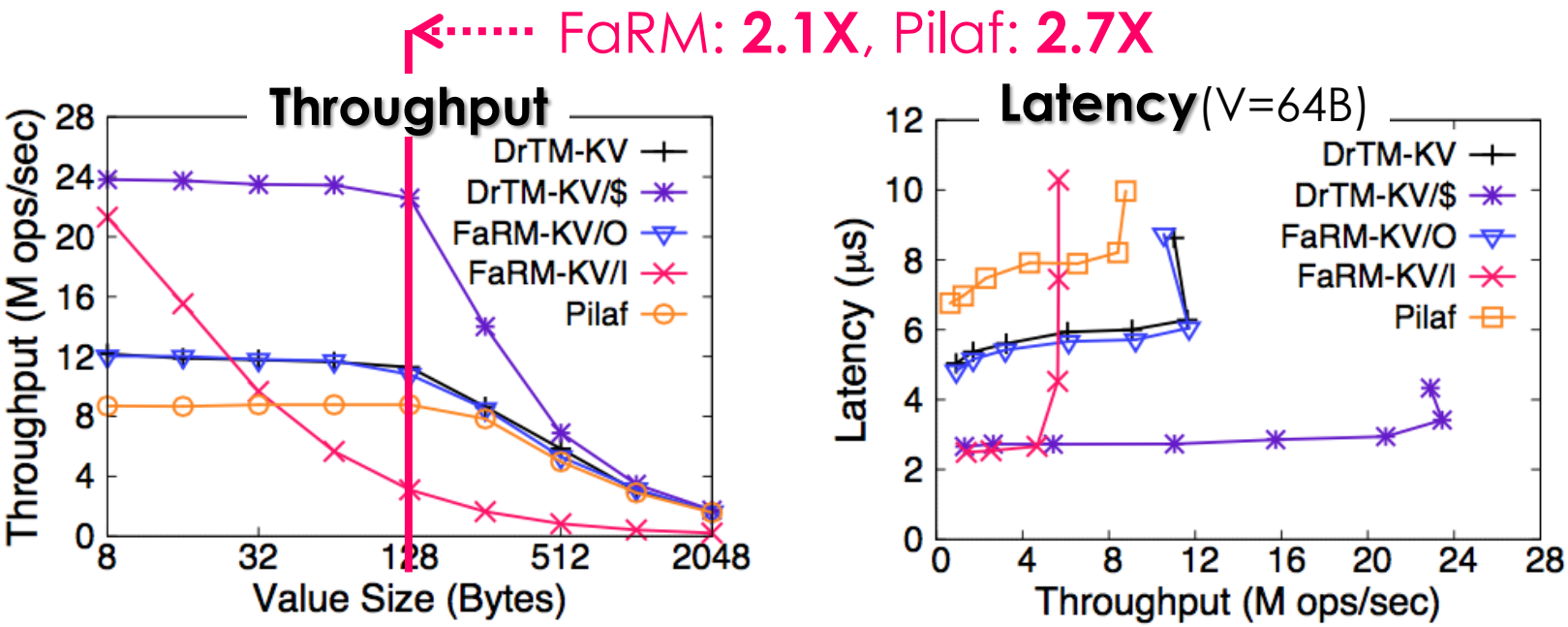


The average lookup cost = **0.178**

20 million kv-pairs (>40 GB), 20MB cache (from empty), 8 client threads, skewed workload (Zipf θ=0.99)

Performance of DrTM-KV

- ✓ DrTM-KV w/o caching provides a comparable performance
- ✓ DrTM-KV w/ caching (DrTM-KV/$) can achieve both lowest latency (3.4 µs) and highest throughput (23.4 Mops/sec)



Setting: 1 server and 5 clients (up to 8 threads), 20 million k/v pairs
peak throughput of random RDMA READ ≈ 26 Mops/sec

Performance of Caching