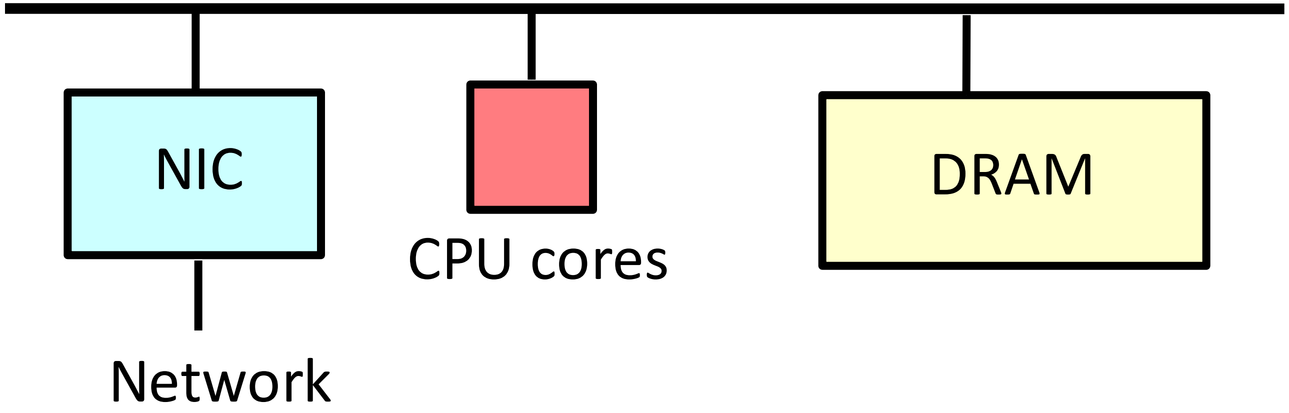
DrTM: Fast In-memory Transaction Processing using RDMA and HTM

Xingda Wei, Jiaxin Shi, Yanzhe Chen, Rong Chen, and Haibo Chen
Institute of Parallel and Distributed Systems
Shanghai Jiao Tong University

Presented by Henggang Cui

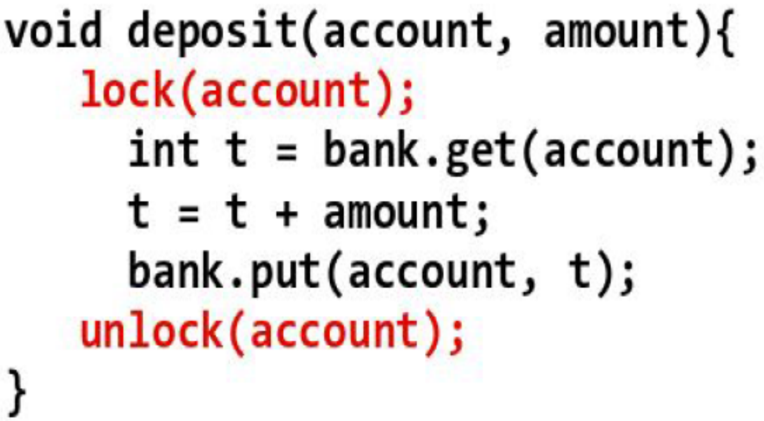
RDMA: Remote direct memory access

- Cross-machine accesses with high speed, low latency, and low CPU overhead
  - Some advanced NICs
  - Direct access to the DRAM of a remote machine
  - By passing remote CPU and OS kernel



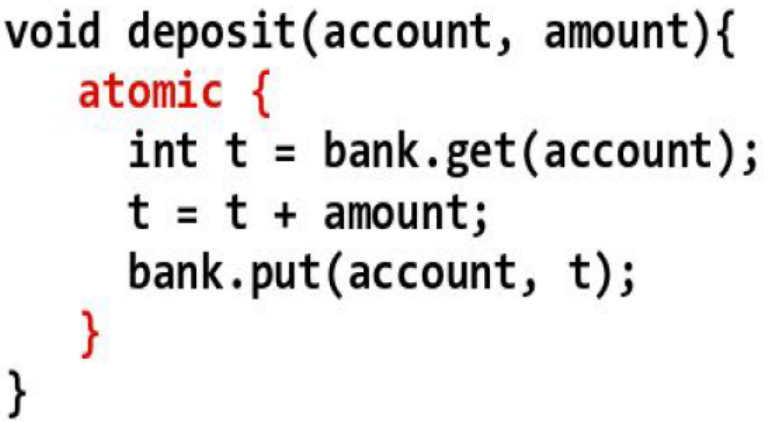
HTM: Hardware transactional memory

Locking:



```
void deposit(account, amount){
  lock(account);
    int t = bank.get(account);
    t = t + amount;
    bank.put(account, t);
  unlock(account);
}
```

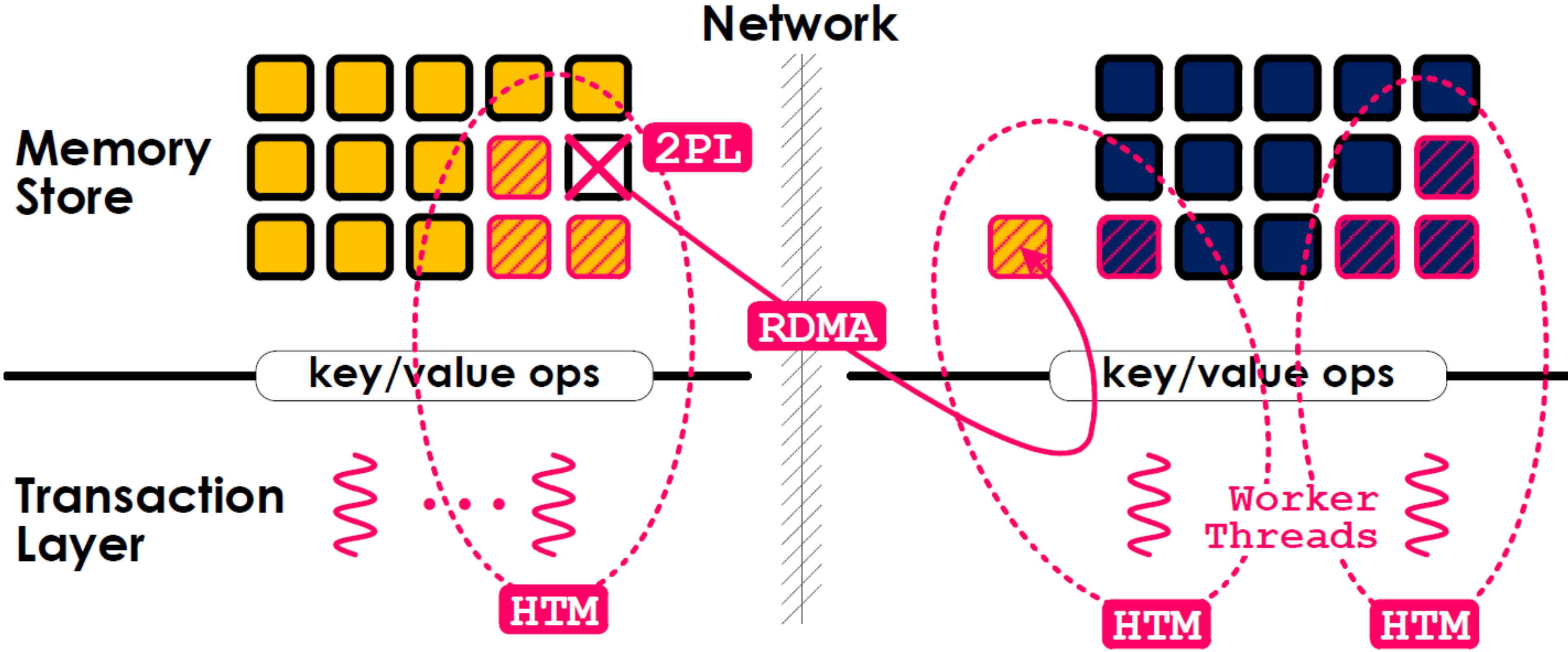
Transactional memory:



```
void deposit(account, amount){
  atomic {
    int t = bank.get(account);
    t = t + amount;
    bank.put(account, t);
  }
}
```

- One way of synching shared memory among threads
  - No locking
  - Access and abort on conflicts
  - Can be understood as optimistic concurrency control

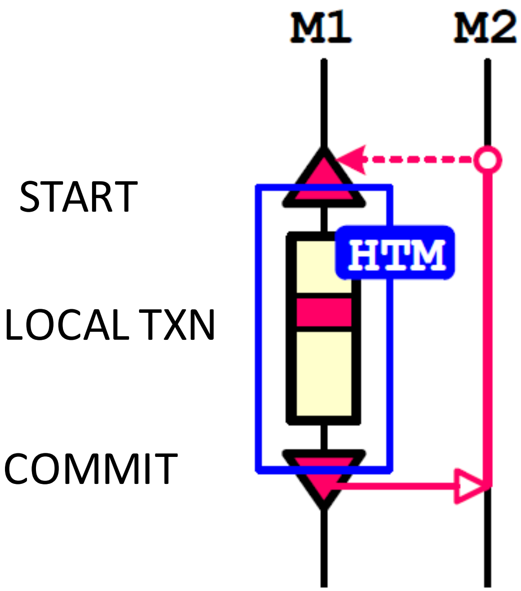
DrTM overview



Transaction layer

- Supporting distributed transactions
  - HTM within a single machine
  - Two-phase locking for accessing remote records

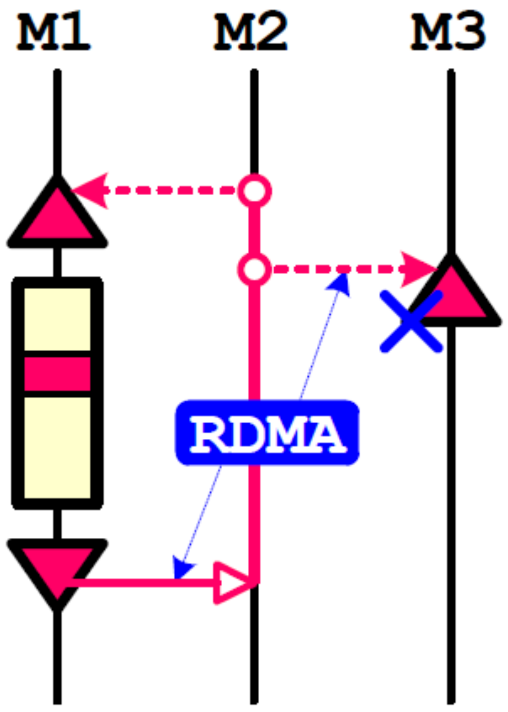
Transactions



(a)

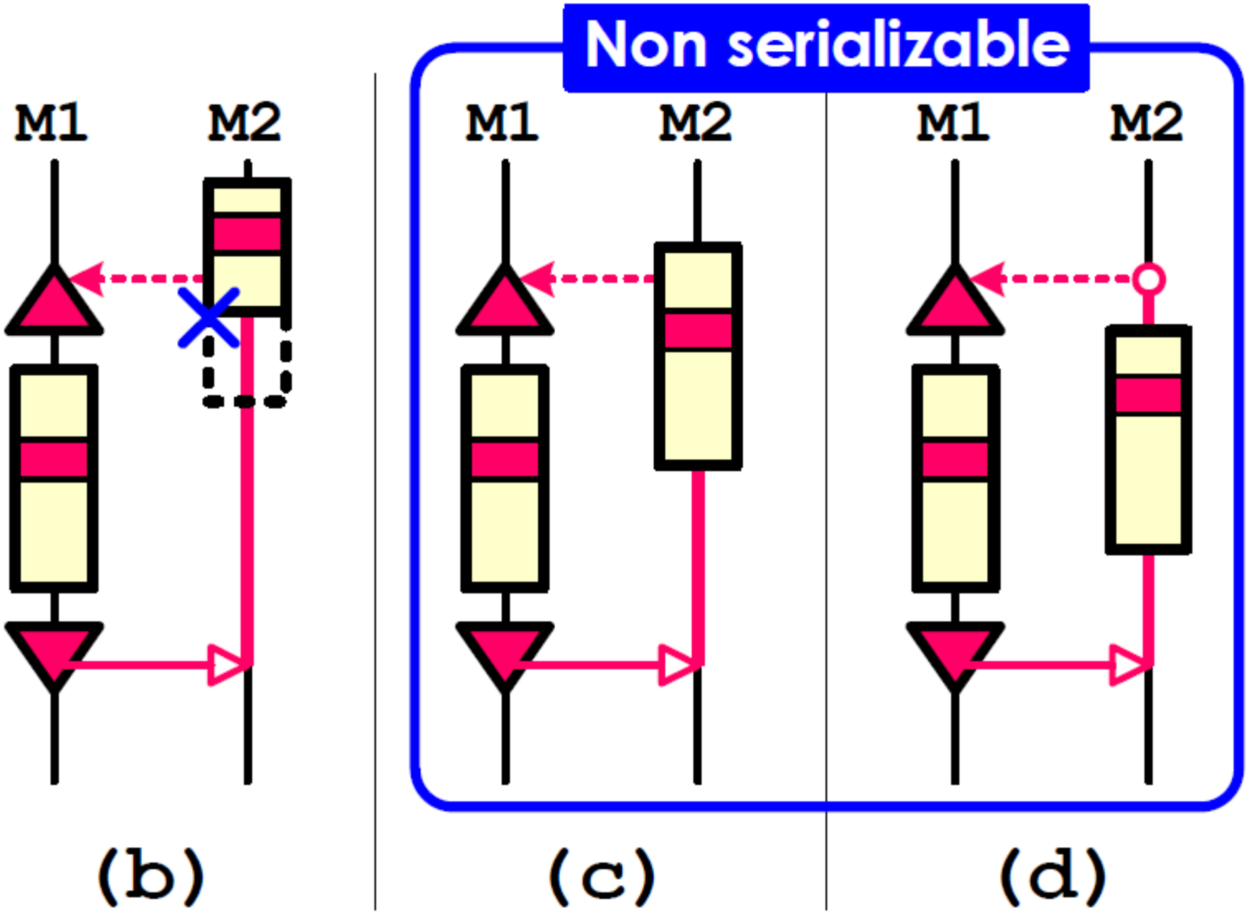
```
START(remote_writeset, remote_readset)
   //lock remote key and fetch value
   foreach key in remote_writeset
     REMOTE_WRITE(key)
   end_time = now + duration
   foreach key in remote_readset
     end_time = MIN(end_time,
                    REMOTE_READ(key, end_time)
   XBEGIN() //HTM TX begin          HTM Transaction

READ(key)
   if key.is_remote() == true
     return read_cache[key]
   else return LOCAL_READ(key)

WRITE(key, value)
   if key.is_remote() == true
     write_cache[key] = value
   else LOCAL_WRITE(key, value)

COMMIT(remote_writeset, remote_readset)
   //confirm all leases are still valid
   if !VALID(end_time)
     ABORT() //ABORT: invalid lease
   XEND() //HTM TX end
   //write back value and unlock remote key
   foreach key in remote_writeset
     REMOTE_WRITE_BACK(key, write_cache[key])
```

Coordinate with other remote txns



(e)

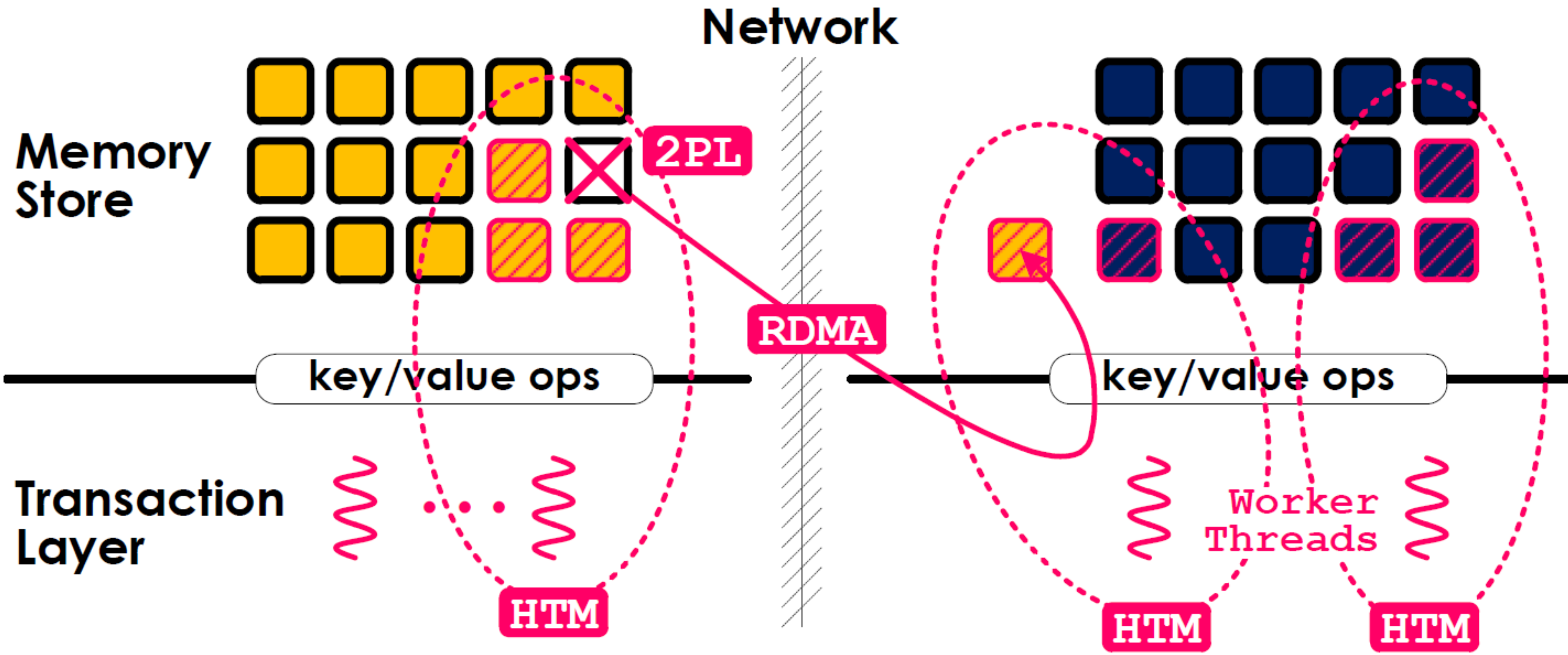
Coordinate local and remote txns



Lease-based shared lock

- They use lease-based shared lock
  - To allow concurrent remote reads
  - Remote read acquires a lease
  - Local and remote write will check the leases
  - And abort itself when the lease is not expired

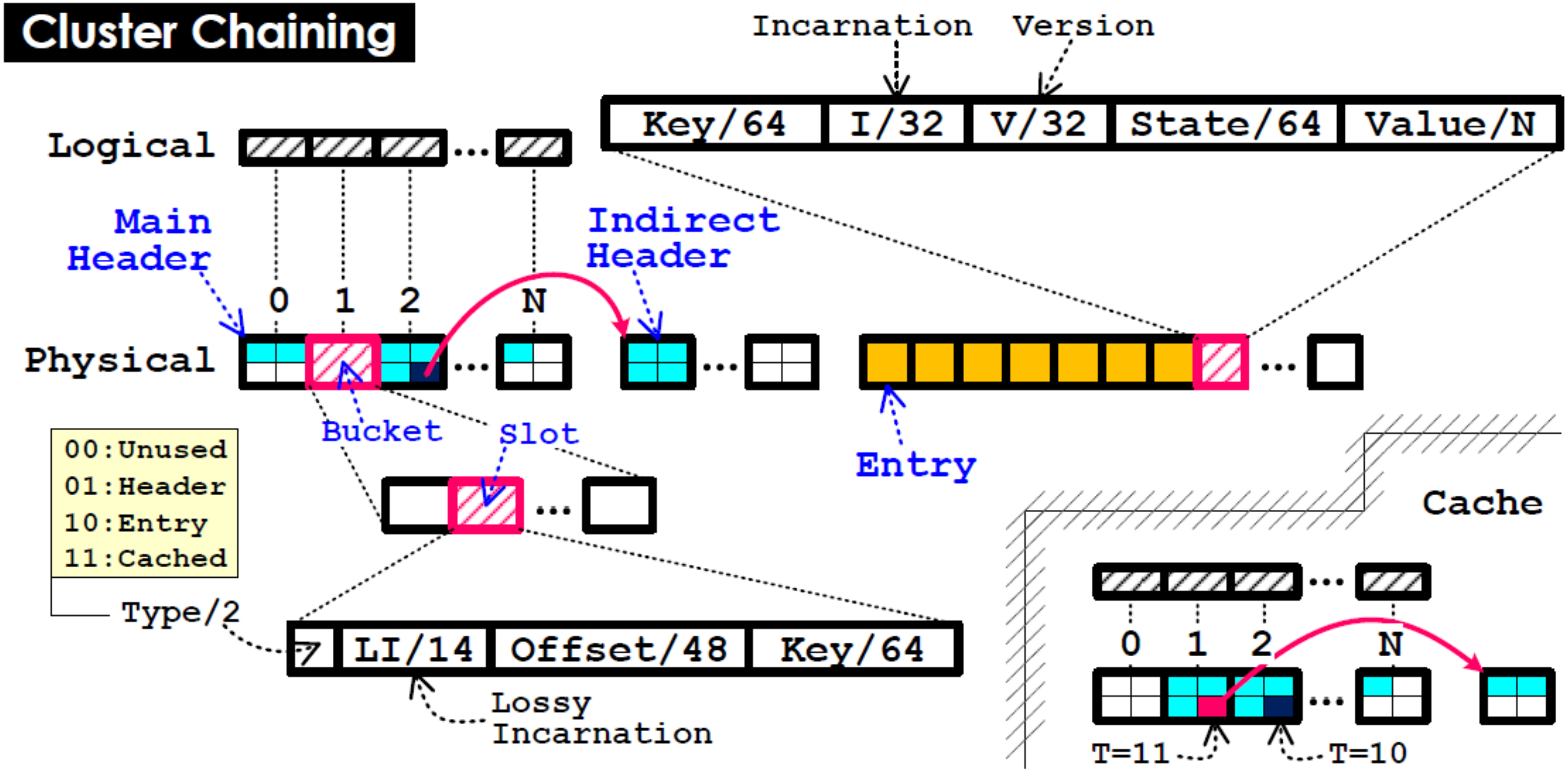
DrTM overview



DrTM memory store

- In-memory key-value store for transaction layer
  - W/ highly optimized hash table based on RDMA and HTM
- They use cluster chaining, as opposed to
  - Cuckoo hashing in Pilaf
  - Hopscotch hashing in FaRM
- They do one-side RDMA for both READ and WRITE

DrTM’s cluster chaining



Caching

- They cache locations instead of values
  - No need for invalidation or synchronization on cache
  - Cached entry location can be shared by threads
  - Size of cached data is smaller

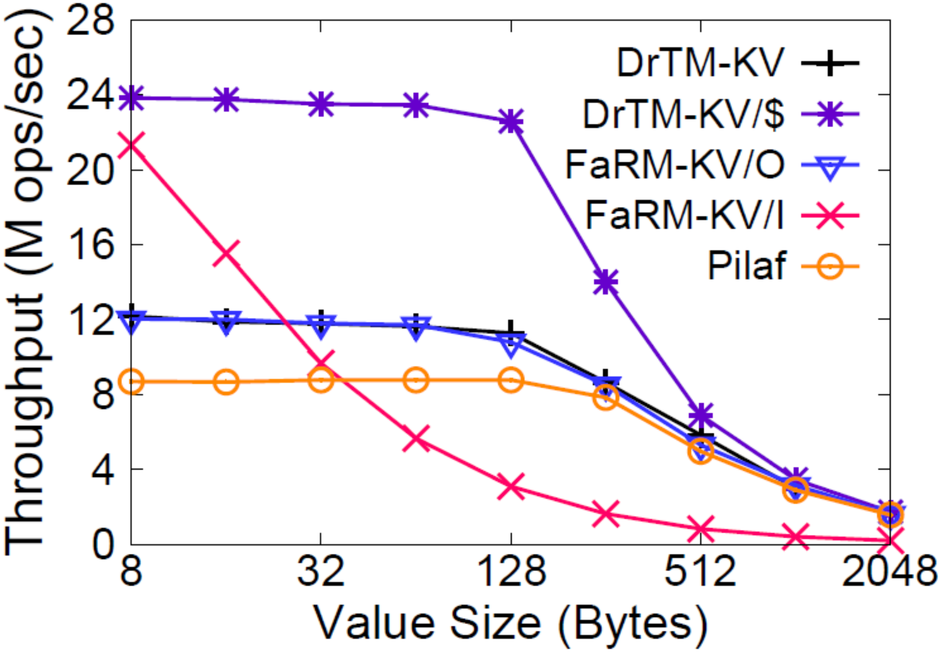
Evaluation

- Experimental setup
  - 6-node cluster
  - Connected by Mellanox ConnectX-3 56Gbps IB
  - Each machine has two 10-core Intel Xeon processors and 64GB of DRAM

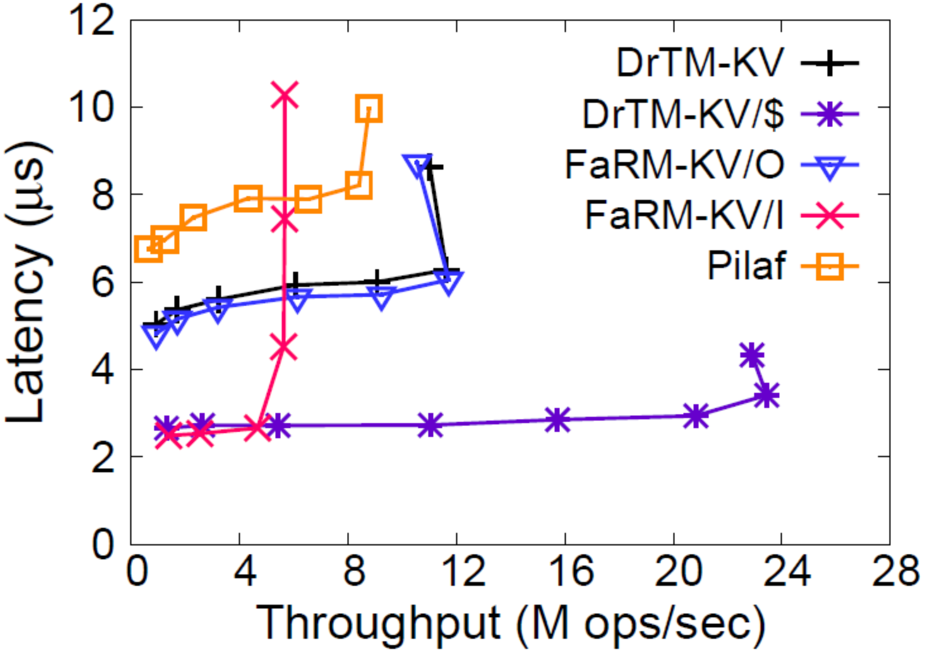
DrTM memory store performance

Dataset: YCSB uniform distribution

Throughput



Latency

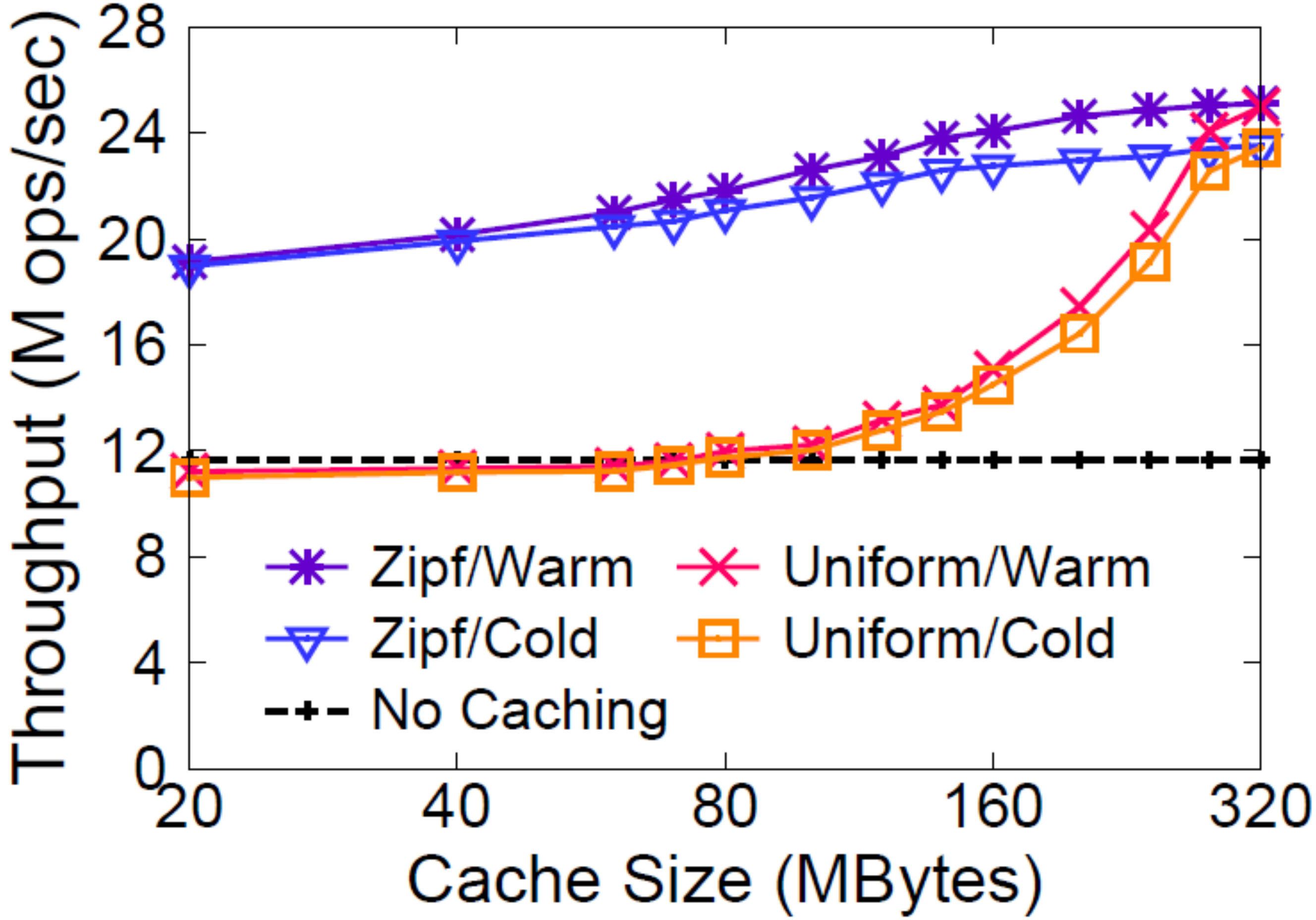


DrTM-KV/$: DrTM-KV with caching

FaRM-KV/I: FaRM-KV that puts key-value pairs **inside** header slots

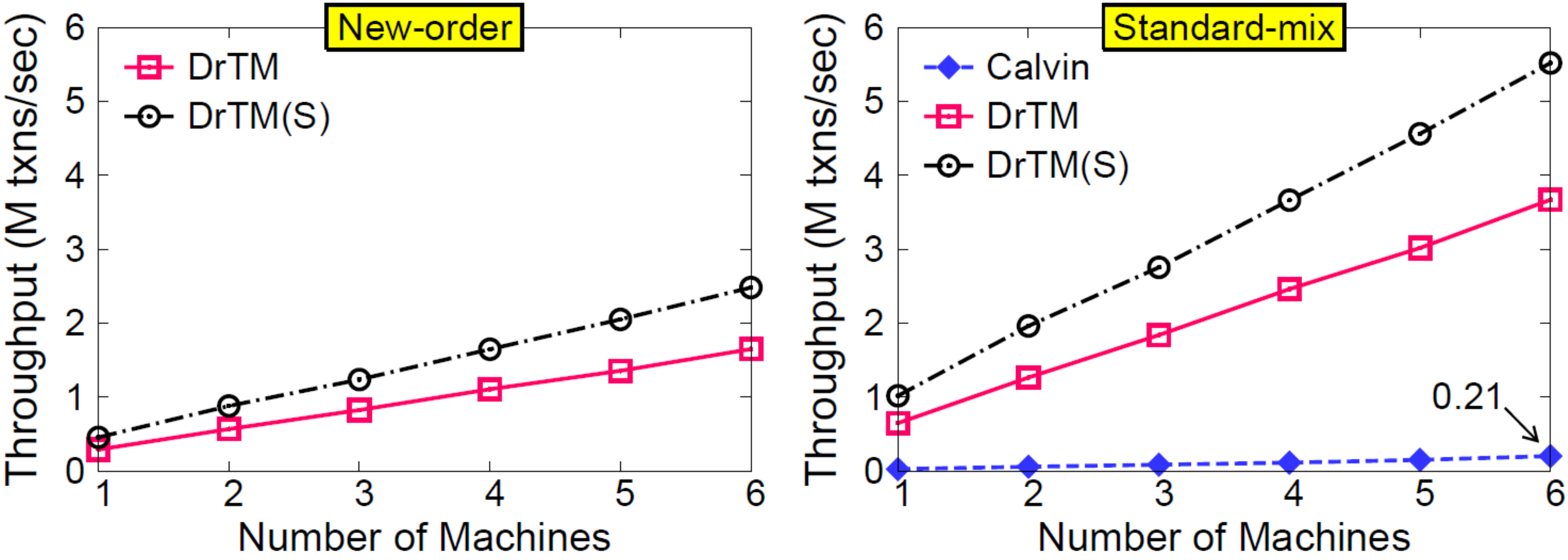
FaRM-KV/O: FaRM-KV that puts key-value pairs **outside** header slots

DrTM memory store performance



DrTM overall performance

Dataset: TPC-C



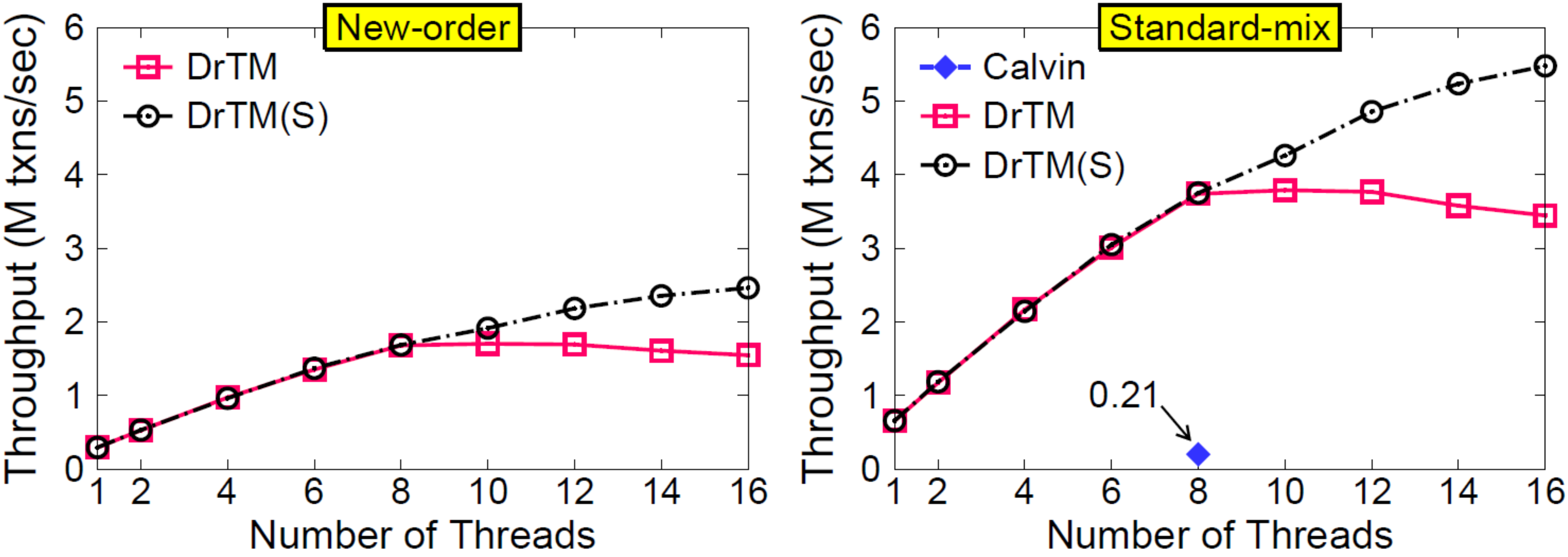
DrTM(S): logical node with 8 worker threads on each socket of a machine

DrTM is 18x faster than Calvin

(Cui: but they use different number of threads, see next slide)

DrTM overall performance

Using 6 machines



DrTM: logical node with 8 worker threads on each socket of a machine

Calvin hard-codes number of threads to 8

Conclusions

- Fast Distributed TxNs using RDMA + HTM
- HTM/RDMA friendly hash table