IPADS, SJTU, China http://ipads.se.sjtu.edu.cn/drtm.html

TALK @ MON Oct 5
12:00 SESSION 2
Distributed Transactions

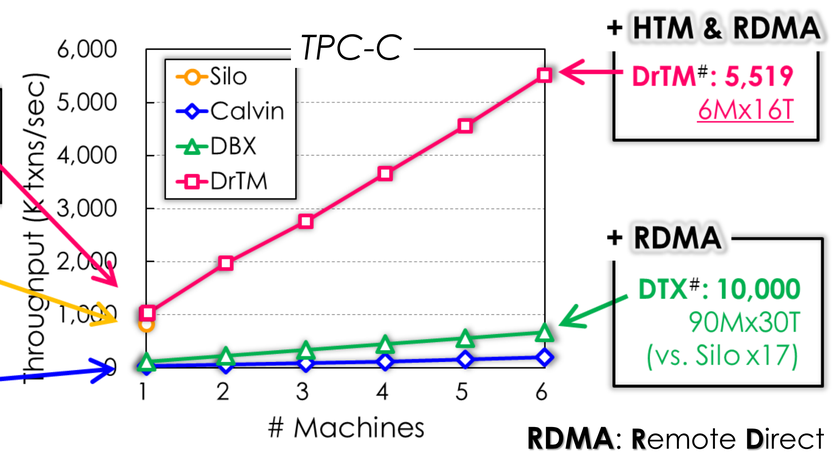
DrTM: Fast In-memory Transaction Processing using RDMA and HTM

Xinda Wei, Jiaxin Shi, Yanzhe Chen, Rong Chen and Haibo Chen

In-memory transaction processing system is a key pillar for many systems like order entry, Web service, and stock exchange

Can we build a distributed transaction system scale from single machine using HTM and RDMA

HTM: **H**ardware **T**ransaction **M**emory
Allow a group of load & store instr. to execute in an atomic, consistent and isolated (ACI) way



Single machine
- High performance
- Not scale-out

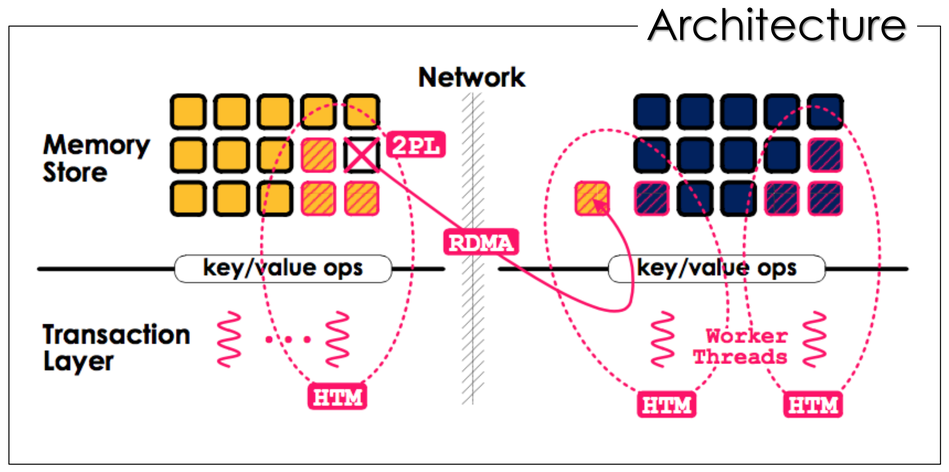
Distributed system
- Good scalability
- Poor performance

RDMA: **R**emote **D**irect **M**emory **A**ccess
Provide cross-machine accesses with high speed, low latency & low CPU overhead

Each: 20-core Intel E5-2650 w/o HT, Mellanox ConnextX-3 56Gbps IB, 1 warehouse/T
$ Each: 16-core Intel E5-2650 w/ HT, Mellanox ConnectX-3 56Gbps IB, 240 warehouse/M

DrTM: In-memory TX processing system
- Target: OLTP workloads
- Two independent components:
  Transaction layer & **memory store**
- A partitioned global address space

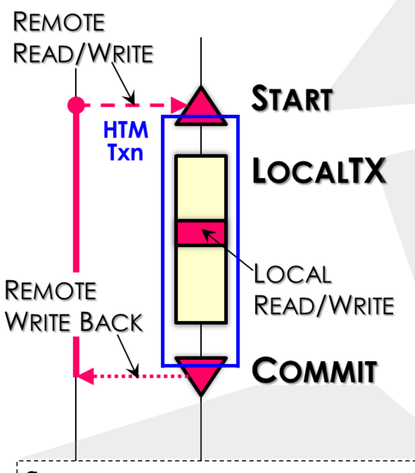
Architecture



Fully leverage processor & networking features
Local TX: HTM & Distributed TX: 2PL + HTM

Transaction Layer

DrTM's Transaction:
START+LOCALTX+COMMIT




```
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
```

```
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)
```

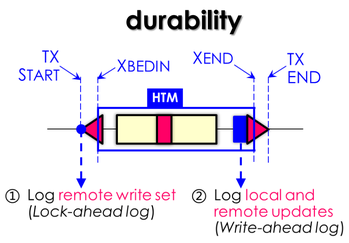
```
COMMIT(remote_writeset,remote_readset)
  //confirm all leases are still valid
  if !VALID(end_time)
    ABORT() //ABORT: invalid lease
  XEND() //HTM TX end
  //write back value & unlock remote key
  foreach key in remote_writeset
    REMOTE_WRITE_BACK(key,write_cache[key])
```

```
LOCAL_READ(key)
  if states[key].w_lock == W_LOCKED
    ABORT() //ABORT: write locked
  else // no conflict w/ remote write
    return values[key]
```

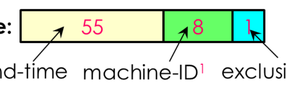
```
LOCAL_WRITE(key, value)
  if states[key].w_lock == W_LOCKED
    ABORT() //ABORT: write locked
  else
    if EXPIRED(END_TIME(states[key]))
      values[key] = value
    else
      ABORT() //ABORT: read locked
```

```
REMOTE_WRITE_BACK(key, value)
  RDMA_WRITE(key, value)
  //unlock
  RDMA_CAS(key, LOCKED(m_id), INIT)
```

durability

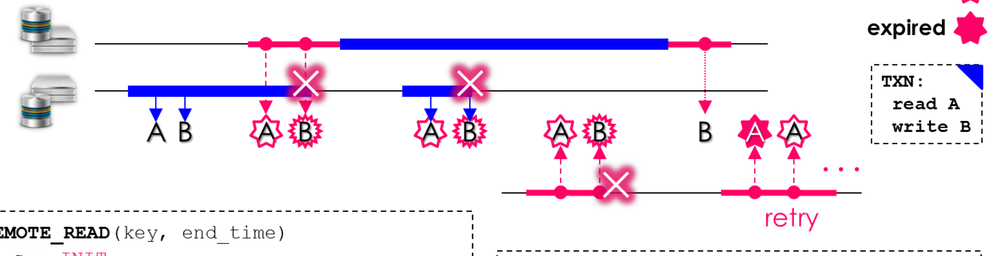


exclusive & shared lock




```
000...000₂ unlocked
000...yy1₂ exclusive locked
xxx...000₂ shared locked
  EXPIRED: if now > end-time + DELTA
  INVALID: if now < end-time - DELTA
```

DELTA is used to tolerate the time bias among machines




```
REMOTE_READ(key, end_time)
  _s = INIT
L:s = RDMA_CAS(key, _s, R_LEASE(end_time))
  if s == _s //SUCCESS: init
    read_cache[key] = RDMA_READ(key)
    return end_time
  else if s.w_lock == W_LOCKED
    ABORT() //ABORT: write locked
  else
    if EXPIRED(END_TIME(s))
      _s = s
      goto L //RETRY: correct s
    else //SUCCESS: unexpired leased
      read_cache[key] = RDMA_READ(key)
      return s.read_lease
```

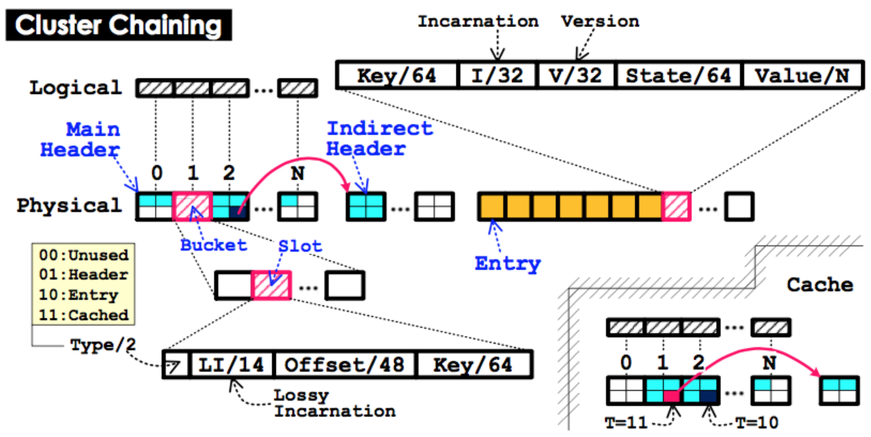
```
REMOTE_WRITE(key)
  _s = INIT
L:s = RDMA_CAS(key, _s, LOCKED(m_id))
  if s == _s //SUCCESS: init
    write_cache[key] = RDMA_READ(key)
  else if s.w_lock == W_LOCKED
    ABORT() //ABORT: write locked
  else
    if EXPIRED(END_TIME(s))
      _s = s
      goto L //RETRY: correct s
    else
      ABORT() //ABORT: unexpired leased
```

Memory Store

HTM/RDAM-friendly Hash Table

	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

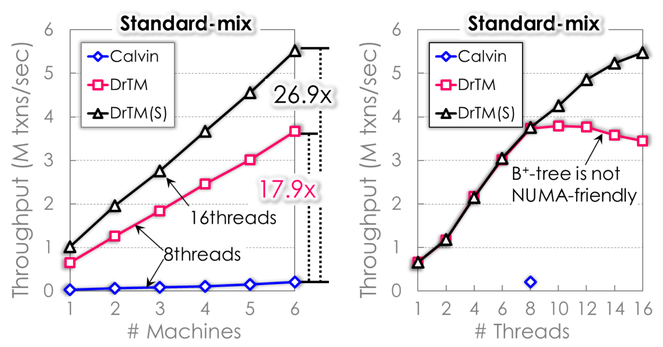
Cluster Chaining



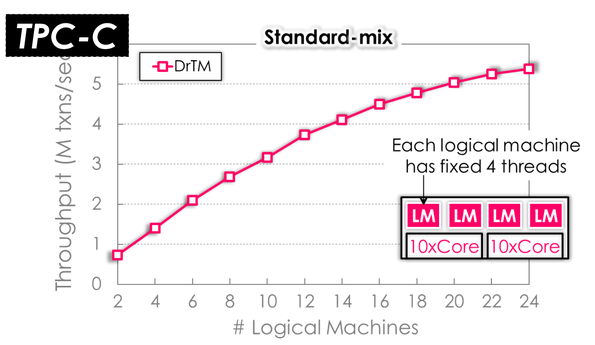
RDAM-friendly, location-based and host-transparent cache

Evaluation

DrTM(S): run a separate logical node on each socket



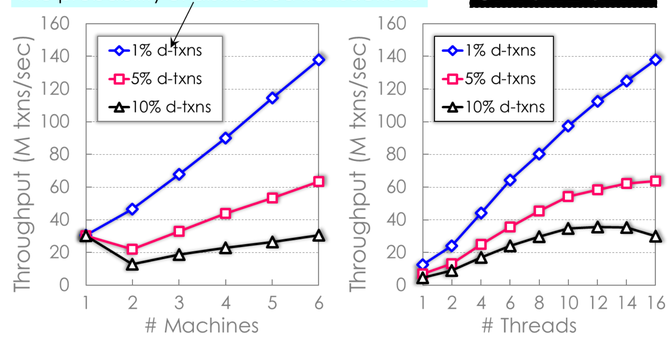
TPC-C



NOTE: the interaction btw. two logical nodes sharing the same machine still uses our RDMA-friendly 2PL protocol

The probability of distributed transactions

SmallBank



		w/o logging	w/ logging
Standard-mix (txns/sec)		3,670,355	3,243,135
New-order (txns/sec)		1,651,763	1,459,495
Latency (µs)	**average**	13.26	15.02
	50%	6.55	7.02
	90%	23.67	30.45
	99%	86.96	91.14
Capacity Abort Rate (%)		39.26	43.68
Fallback Path Rate (%)		10.02	14.80

Setting: 6 machines with 8 threads

A 6-node Cluster: two 10-cores, RTM-enabled Intel E5-2650 w/o HT
Mellanox ConnectX-3 MCX353A 56Gbps InfiniBand NIC w/ RDMA