

In-memory Transactions

A Perspective from Systems Software



HAIBO CHEN

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

Institute of Parallel and Distributed Systems
Shanghai Jiao Tong University, China

Joint work with Rong, Xinda, Jiaxin, Yanzhe, Heng, Mingkai, etc.@IPDADS, the wukong work is also with Fefei@Utah

Tape is Dead
Disk is Tape
Flash is Disk
RAM Locality is King

Jim Gray
Microsoft
December 2006

In memory of Jim Gray



Tape is Dead

Disk is Tape

Flash is Disk

RAM is Flash?


Cache Locality/Parallelism is King?

Transaction: Key Pillar for Many Systems

amazon.com
426 items/sec

Alibaba.com
\$9.3 billion/day

Demand Speedy **Distributed transaction**
Over Large Data Volumes

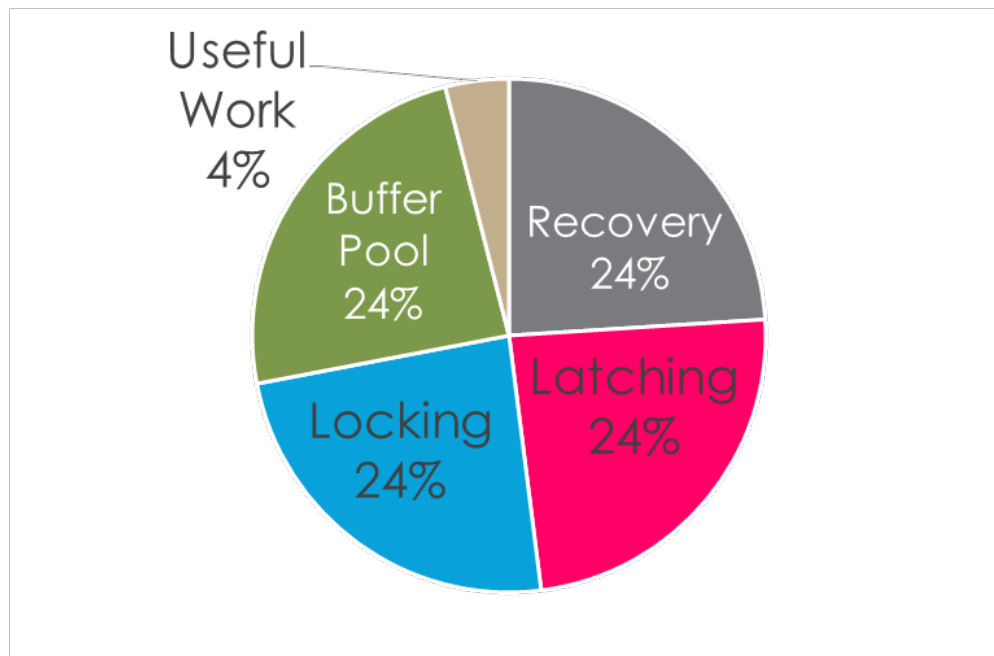

12306
9.56 million
tickets/day

PayPal
11.6 million
payments/day

Conventional DBMSs are Inefficient

Only **4% of wall-clock time** spent on useful data processing, while the rest is occupied with **buffer pools, locking, latching, recovery**.¹

-- Michael Stonebraker



¹ "The Traditional RDBMS Wisdom is All Wrong"

Business Demand – High Throughput



**GLOBAL SHOPPING
FESTIVAL 2016**

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

**“To Be or Not to Be:”
It is a Matter of **Time****

- 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

TPCC world record

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

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



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

How to Do It? Today's Approach

OceanBase

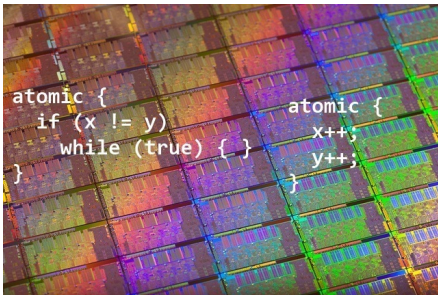
MySQL Cluster



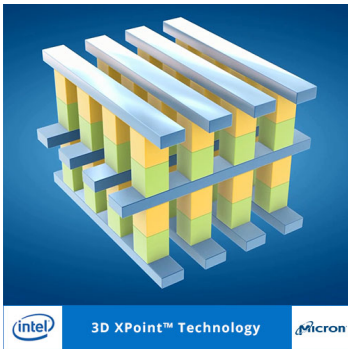
How to Do It? Our Approach

HTM

6 nodes connected with IB
Cost: 73,800\$



NVM



RDMA



A few rack-scale machines

This Talk

101 of HTM and RDMA

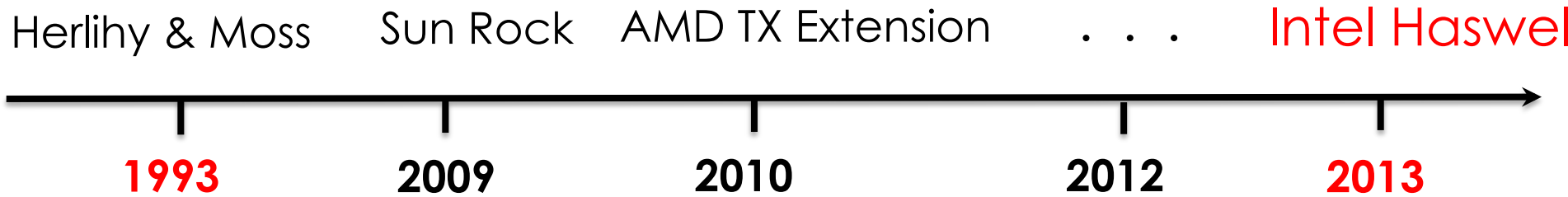
Overall ideas

RDMA-friendly distributed key-value store

Fast distributed transactions using RDMA & HTM

System software support for In-memory Transactions

Hardware Transactional Memory



Massively available

Intel Restricted Transactional Memory

Restricted Transactional Memory (RTM)

- Hardware transactional memory with limitations

Major limitations

- Working set is limited
- System events abort TX

New instruction set

- Xbegin, Xend, Xabort

RTM Usage

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

Handle the
abort event

Deconstructing RTM

RTM prefer read than write

- Asymmetric Read/Write Limits
 - L1-Cache tracks writes
 - An implementation specific structure tracks reads

RTM prefers read before write

- Only eviction of cache lines in write set will abort TX

Transaction exec time affects TX abort

- Timer interrupt unconditionally abort a TX (4ms on 250hz kernel)

RDMA: Remote Direct Memory Access

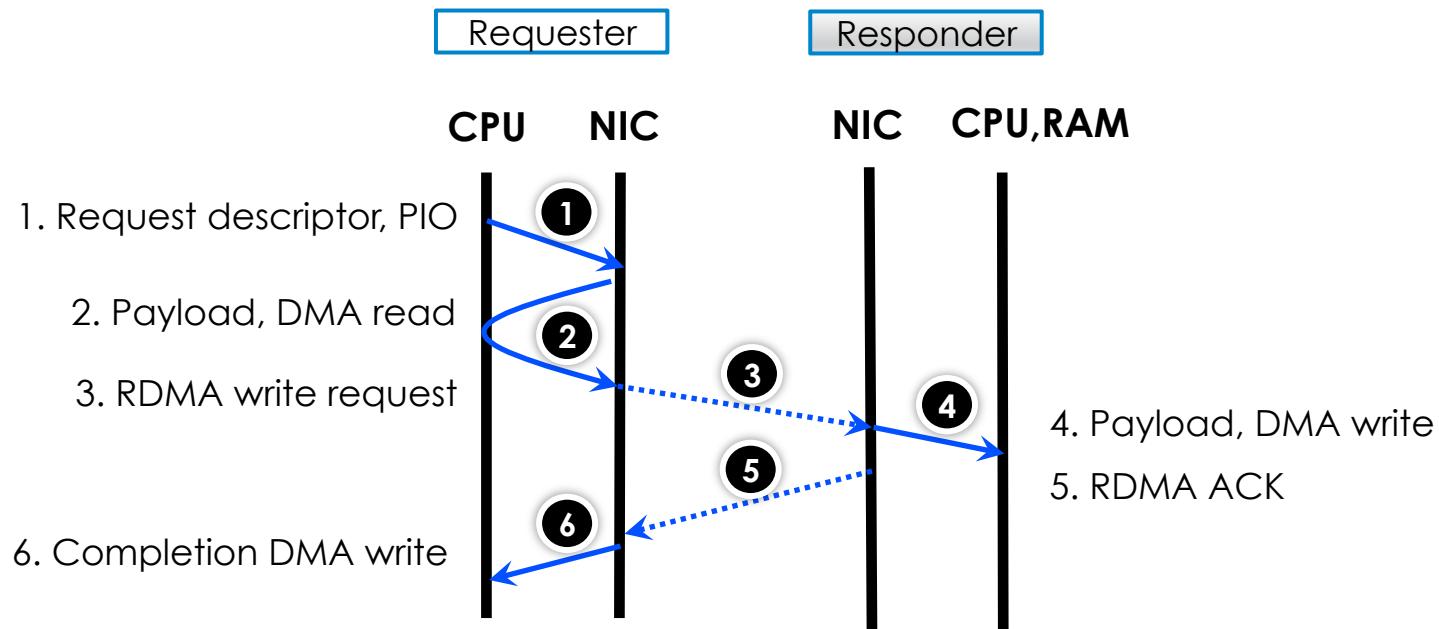
A network feature that allows direct access to the memory of a remote computer

High speed, low latency & low CPU overhead

- Interface: SEND/RECV Verbs, and one-sided RDMA (READ/WRITE/CAS), IPoIB, etc.
- Bypasses OS kernels: Zero copy
- Round-trip time: one-sided/ $\sim 3\mu\text{s}$, verb msg/ $\sim 7\mu\text{s}$, IPoIB/ $\sim 100\mu\text{s}$

One-sided RDMA Primitives

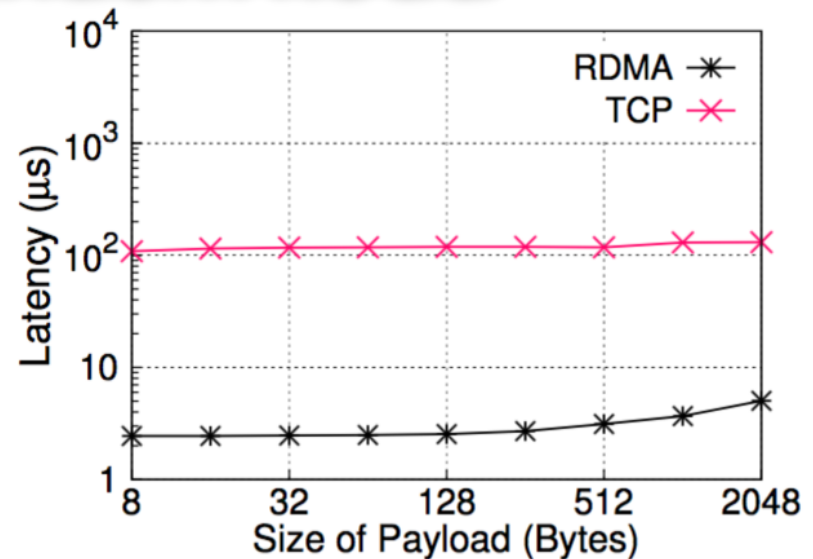
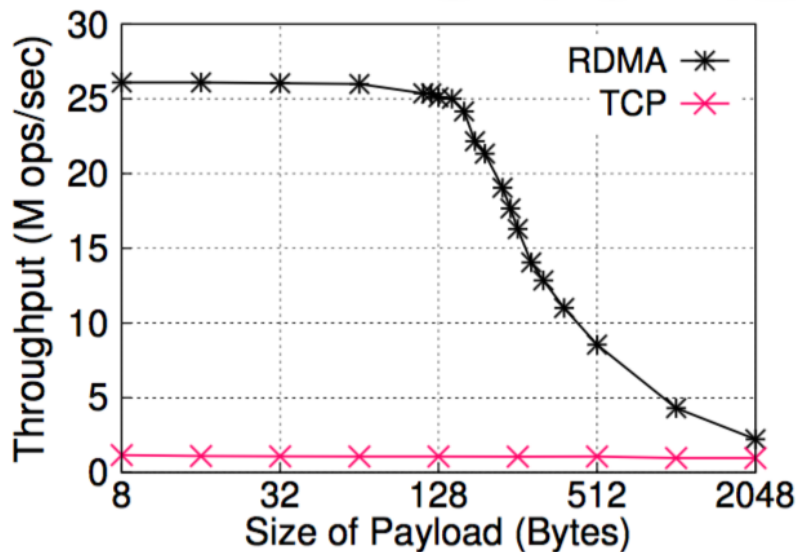
RDMA read, write and CAS Life-cycle of an RDMA write



Credit: Anuj Kalia's SIGCOMM talk

One-sided RDMA Performance

Perf. of Random Read¹



Insensitive to payload size:

High/near constant throughput/Low latency when payload is smaller than a threshold

¹ Mellanox ConnectX-3 MCX353A 56Gbps InfiniBand NIC

Overall Ideas: Combining
Advanced Hardware Features for
In-memory Transactions

Opportunities: (not so) New HW Features

HTM: Hardware Transaction Memory

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

RDMA: Remote Direct Memory Access

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

Rethink the design of **low-COST** scalable in-memory transaction systems

Opportunities with HTM & RDMA

HTM: Hardware **T**ransaction **M**emory

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

**Strong
Atomicity**

RDMA: Remote **D**irect **M**emory **A**ccess

Opportunities with HTM & RDMA

HTM: Hardware Transaction Memory

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

**Strong
Atomicity**

RDMA: Remote Direct Memory Access

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

**Strong
Consistency**

Opportunities with HTM & RDMA

HTM: Hardware Transaction Memory

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

RDMA: Remote Direct Memory Access

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

HTM Strong Atomicity + **RDMA Strong Consistency** \Rightarrow **RDMA ops will abort conflicting HTM TX**

Opportunities with HTM & RDMA

HTM: Hardware Transaction Memory

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

RDMA: Remote Direct Memory Access

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

HTM Strong Atomicity

+

RDMA Strong Consistency

⇒

RDMA ops will abort *conflicting* HTM TX



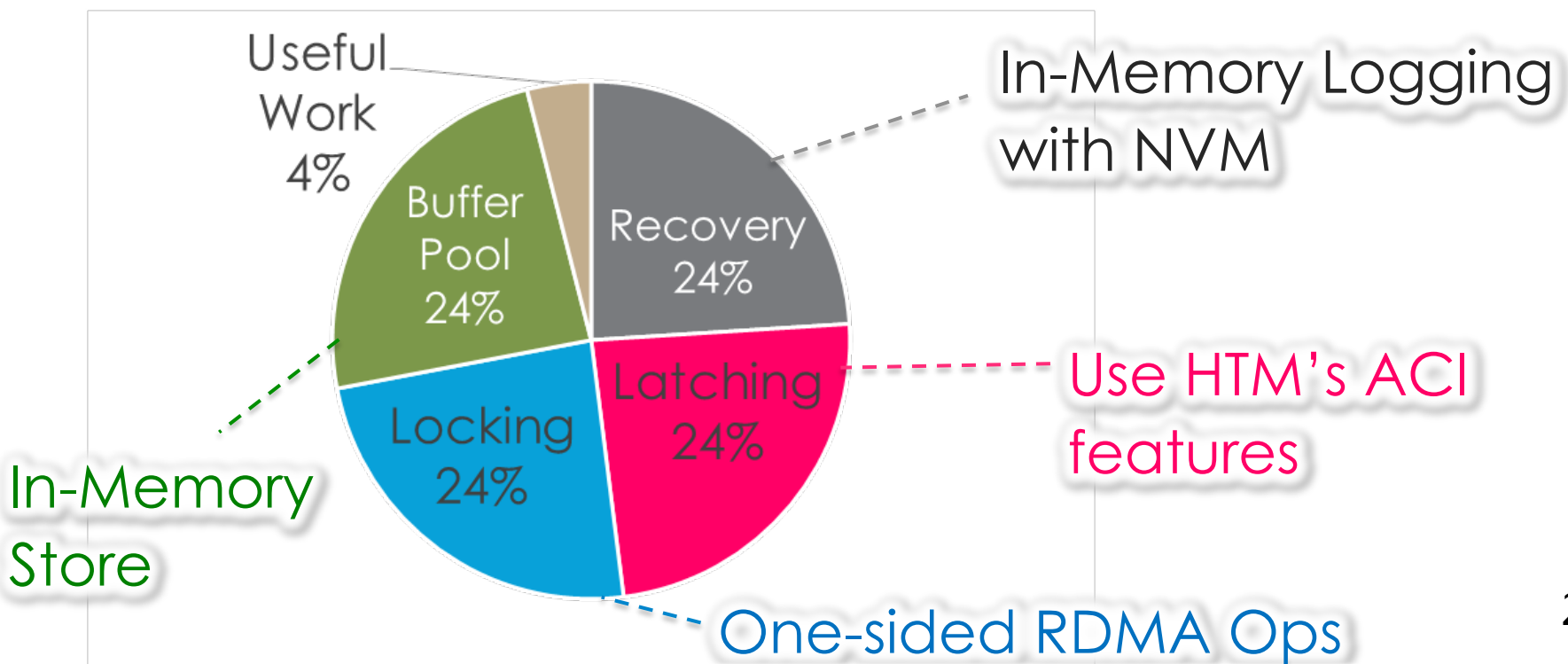
Basis for Distributed TM

Overall Idea

Use **HTM**'s ACI properties for local TX execution

□ DBX (EuroSys'14) DBX-TC (TR'15, TX chopping)

Use one-sided **RDMA** to glue multiple HTM TXs



Recent Work on In-memory TXs



Massive
#Users

DrTM:
SOSP'15
EuroSys'16

Distributed TX

Distributed Query

Wukong:
OSDI'16

DrTM-KV:
SOSP'15

Distributed Key/Value Store

Cocytus:
FAST'16

DBX:
EuroSys'14

Single Machine TX

Single Machine TX

IC3:
SIGMOD'16

Eunomia:
PPoPP'17

OS/VMs

OS/VMs

VPM:
SoCC'16

Prwlock:
ATC'14



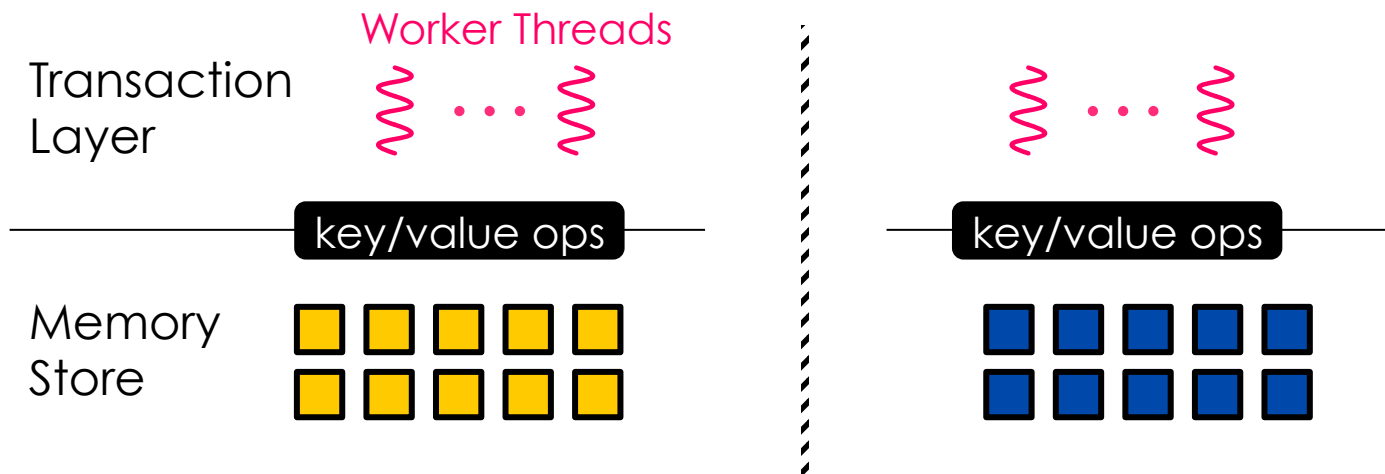
RDMA



Building Fast In-memory Transactions using **RDMA** and **HTM**

DrTM: Distributed TX with HTM & RDMA

- Target: **OLTP** workloads over large volume of data
- Two independent components using HTM&RDMA
 - Transaction layer & memory store (DrTM-KV)**
- Low **COST** distributed TX
 - Achieve over **5.52 million** TXs/sec for TPC-C on 6 nodes



Review: Opportunities with HTM & RDMA

HTM: Hardware Transaction Memory

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

RDMA: Remote Direct Memory Access

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

HTM Strong Atomicity + **RDMA Strong Consistency** \Rightarrow **RDMA ops will abort *conflicting* HTM TX**

Basis for Distributed TM

DrTM: Combining HTM with 2PL

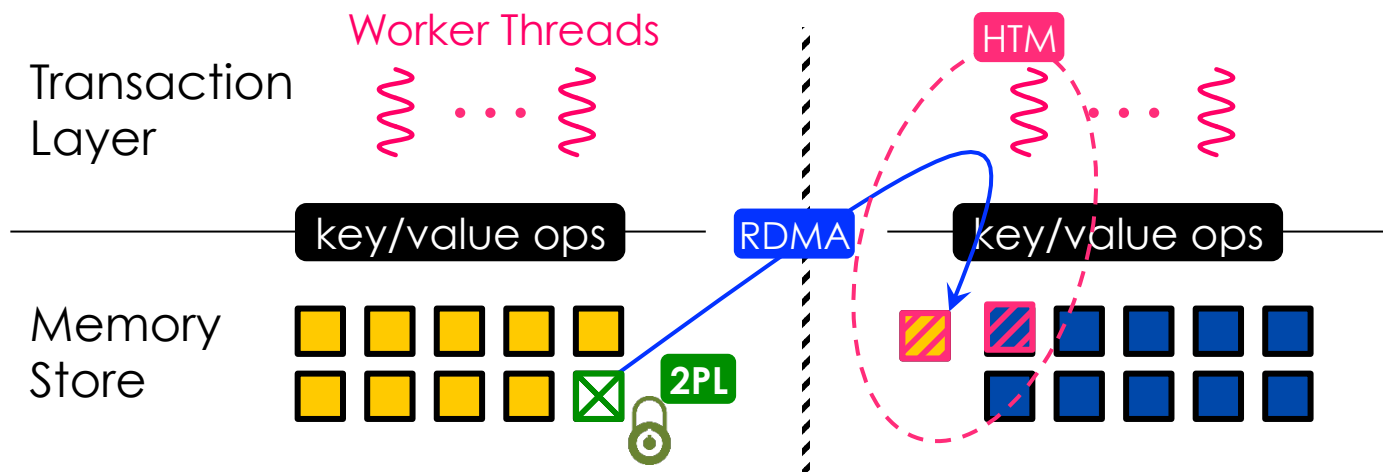
Using RDMA+**2PL** to accumulate all remote records **prior to** accesses in an HTM transaction

- Transform a distributed TX to a **local** TX
- Concurrency control

Local TX vs. Local TX: **HTM**

Distributed TX vs. Distributed TX: **2PL**

Local TX vs. Distributed TX: **abort local TX**



Challenge: Limit of RDMA Semantics

RDMA provides three communication options

- IPoIB, SEND/RECV and **one-sided RDMA ops**

Good performance (e.g. latency)
and without involving the host CPU

One-sided RDMA has limited interfaces

- READ, WRITE, CAS and XADD

How to support exclusive and shared accesses
in 2PL protocol using one-sided RDMA ops

Exclusive & Shared Lock

RDMA CAS: atomic compare-and-swap

- Similar to the semantic of normal CAS (i.e. local CAS)

1. DrTM's **exclusive** lock

- Always use RDMA CAS to **acquire & release**

2. DrTM's **shared** lock

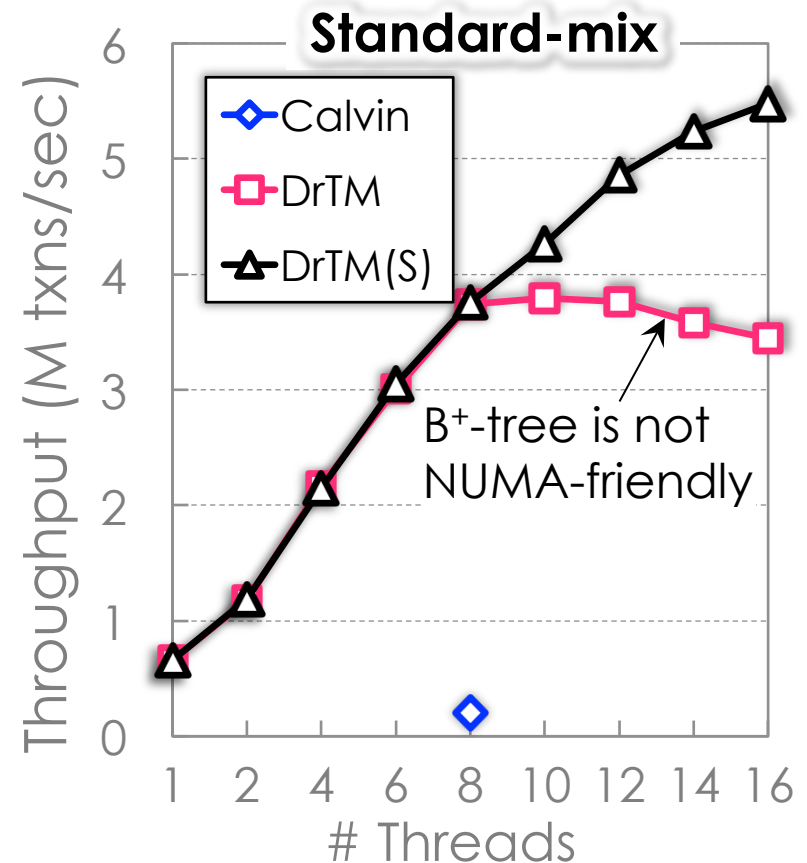
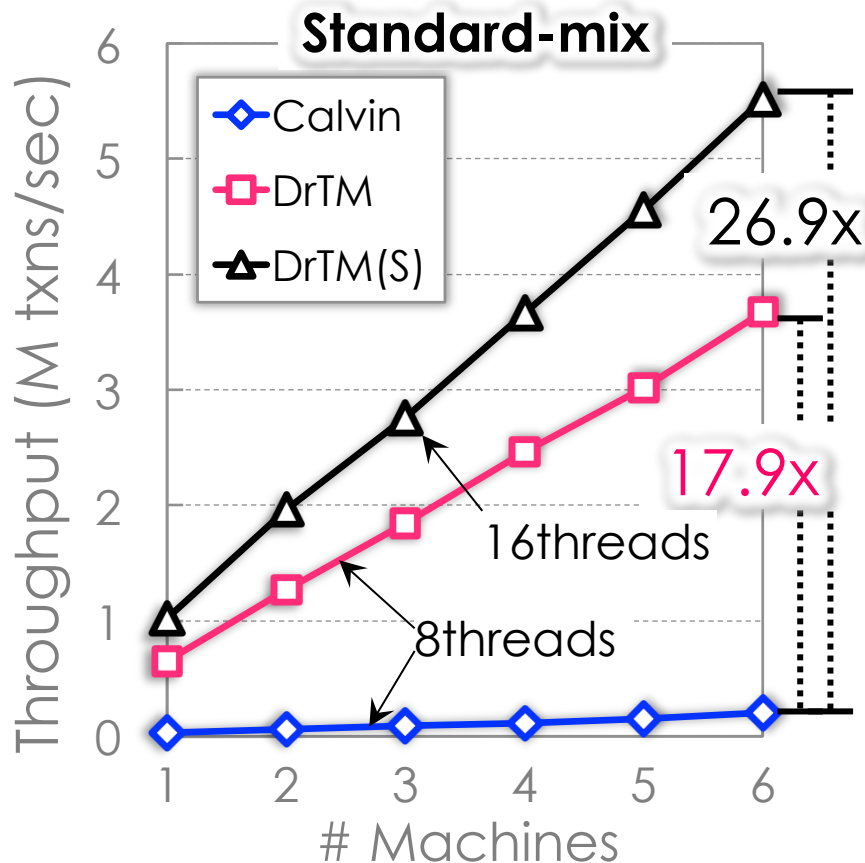
- **Lease-based** shared lock
 - Grant read right to the lock holder in a time period
 - No need to explicit release or invalidate the lock
 - Synchronized time is provided by PTP

Performance on TPC-C

New-order TX
≈ Standard-mix x45%

Throughput:
5.52 millions TX/s

Latency as low as 15.02us



Limitations of DrTM

1. Require advanced knowledge of read/write sets of transactions
2. Preserve durability rather than availability in case of machine failures

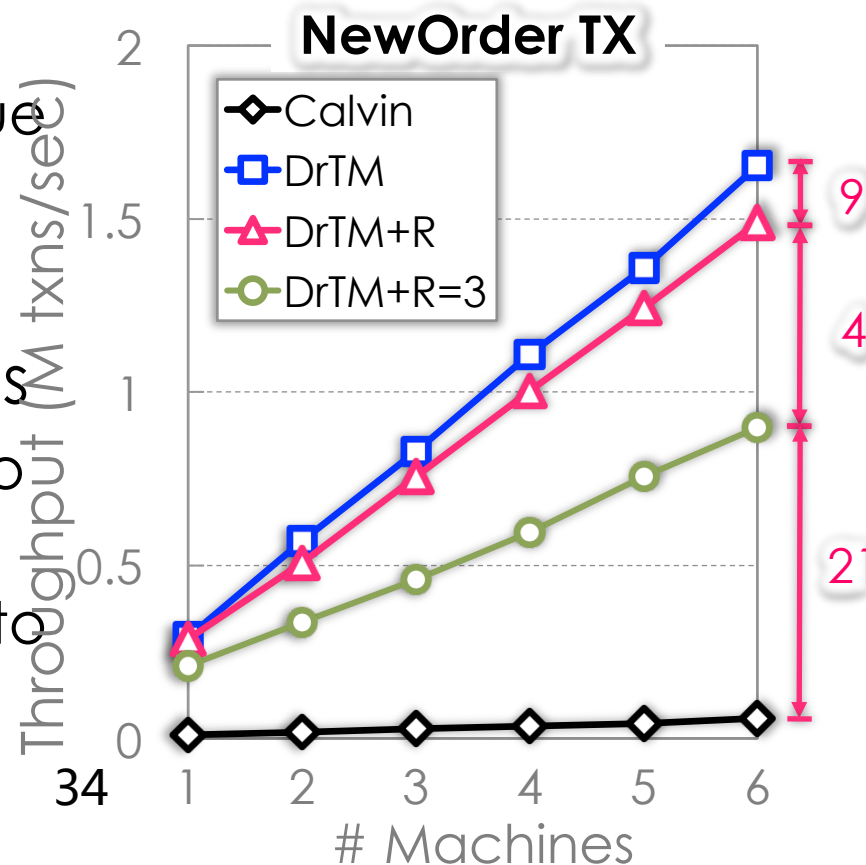
DrTM+R: High Available Distributed TX (EuroSys 2016)

Inherit DrTM's High Performance

- ❑ Use **HTM**'s ACI properties for local TX execution
- ❑ Use one-sided **RDMA** to glue multiple HTM TXs

Overcome DrTM's Limitations

- ❑ Use **Hybrid OCC Protocol** to probe read/write sets
- ❑ Use **Optimistic Replication** to ensure high availability



DrTM-B: Replication-driven Reconfig.

Observation

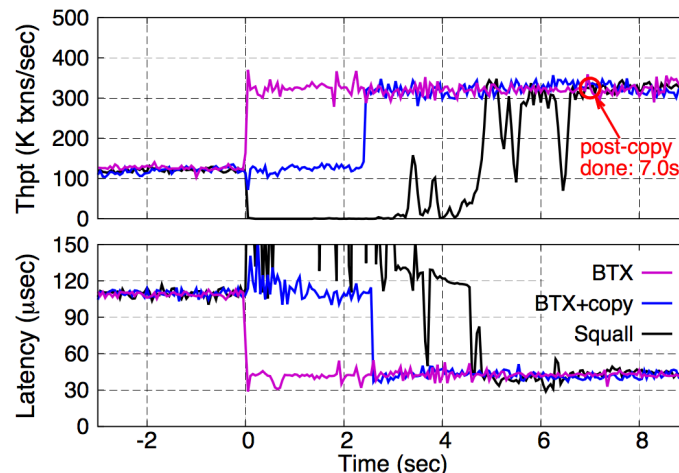
- ❑ TX systems like DrTM-R have >3-way replication

Replication-driven reconfiguration

- ❑ Switch to fault-tolerant replicas when possible to minimize data transfers

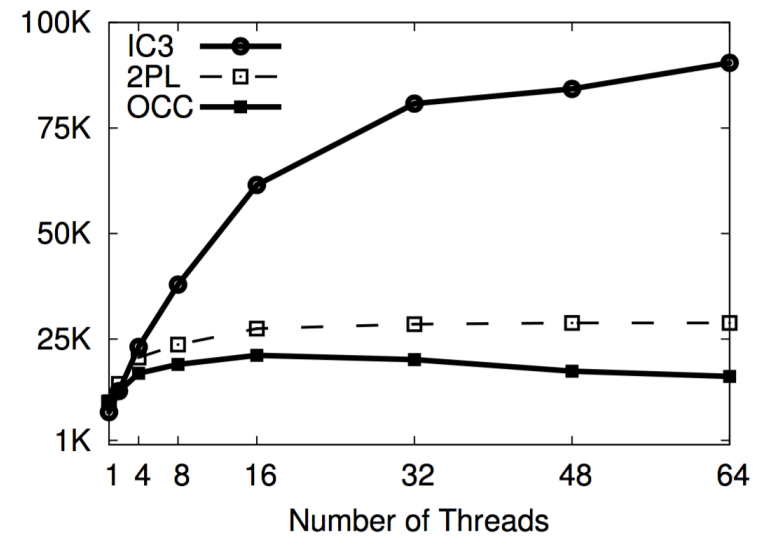
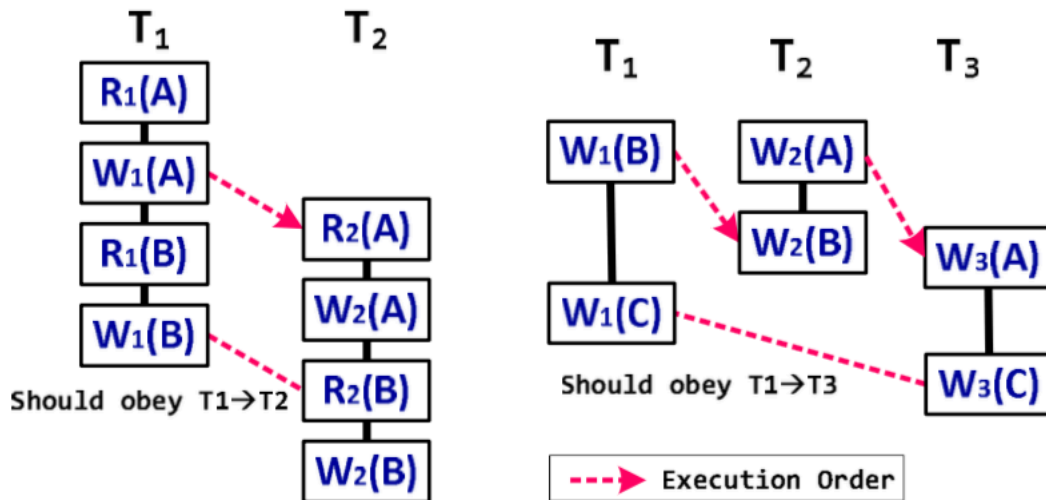
When no idle replicas, construct one on-the-fly

- ❑ Dirty tracking: logs already contain dirtied tuples, reuse log forwarding to sync dirty tuples



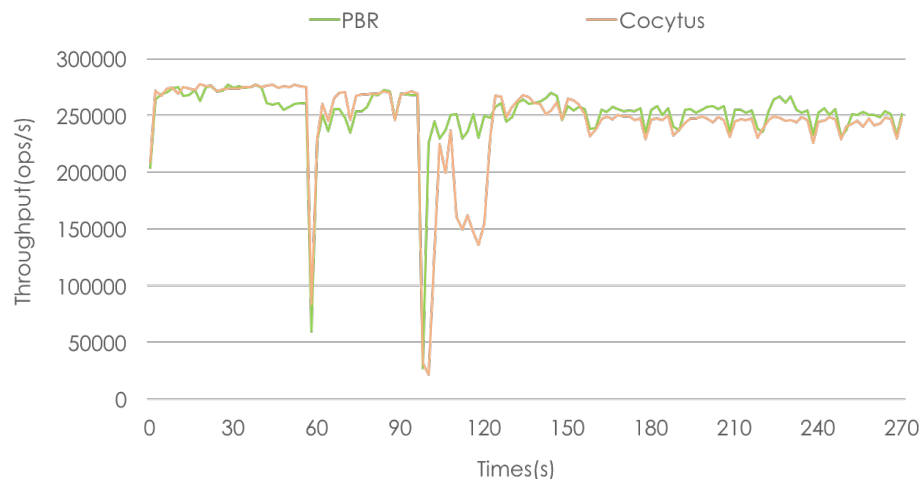
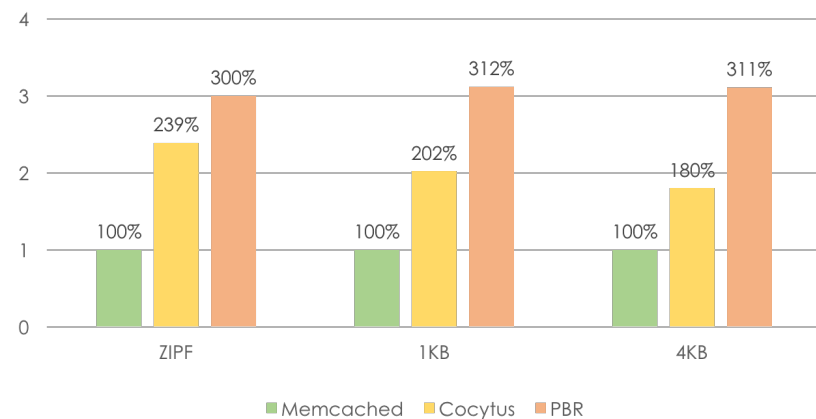
IC3: Refined Concurrency Control (SIGMOD 2016)

- Problem: degraded scalability under high contention
 - OCC: performance collapse
 - 2PL: over-constrained interleaving
- **IC3**: interleaving constrained concurrency control



Cocytus: Reducing Memory Usage (FAST 2016, ToS 2017)

- Erasure coding: high construction time
- Replication: Low memory utilization
- Cocytus: combines erase coding w/ replication
 - Key: primary-backup replication, Value: erasure coding
 - Achieve better memory efficiency w/ low overhead compared with primary-backup replication



Eunomia: Scaling Up B+Tree using HTM (PPoPP 2017)

- HTM-based B+tree:
 - High performance under low contention
 - Collapse under high contention due to excessive aborts
- Eunomia: scalable HTM-B+Tree
 - Splitting large HTM transactions with opportunistic consistency validation
 - Proactively detecting and avoiding true conflicts
 - Adopting adaptive contention control strategy

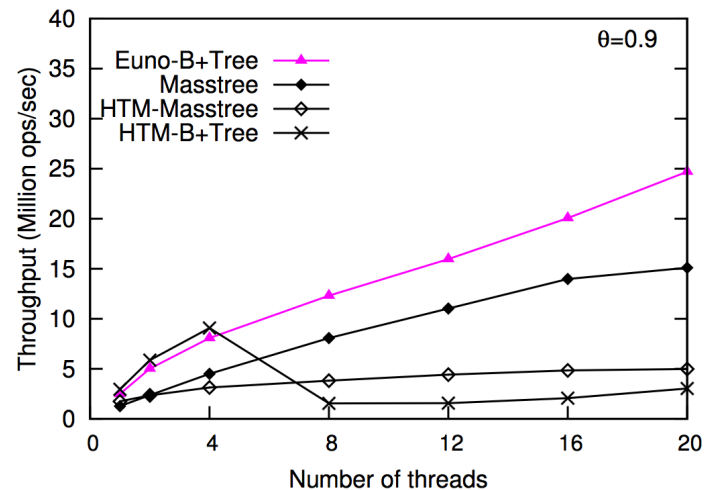
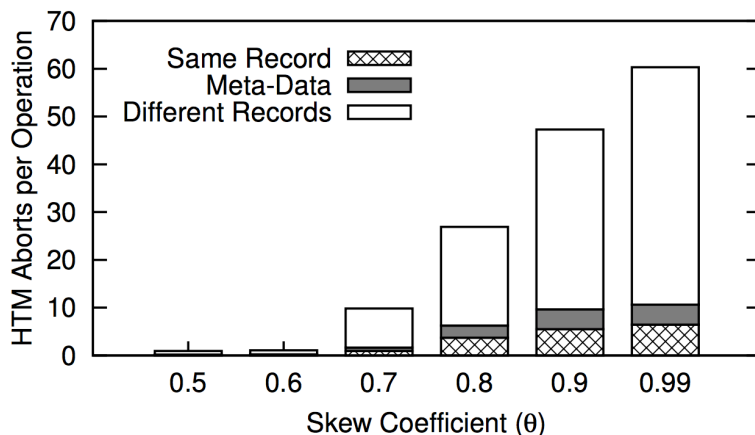
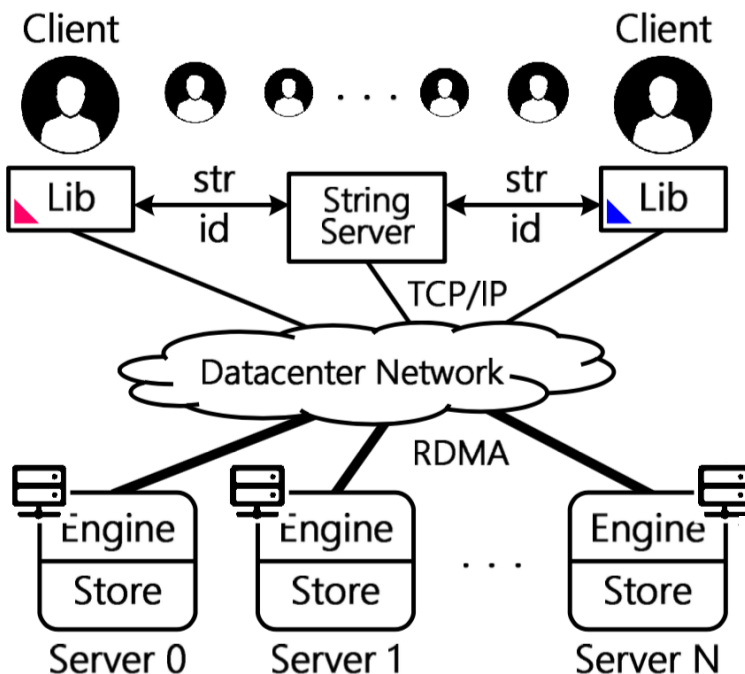


Figure 2: HTM aborts incurred by different reasons.

Distributed Query Processing (OSDI'16)

Wukong: A distributed in-memory RDF store

1. Flexible graph-based model and store
2. Fast and scalable query processing engine

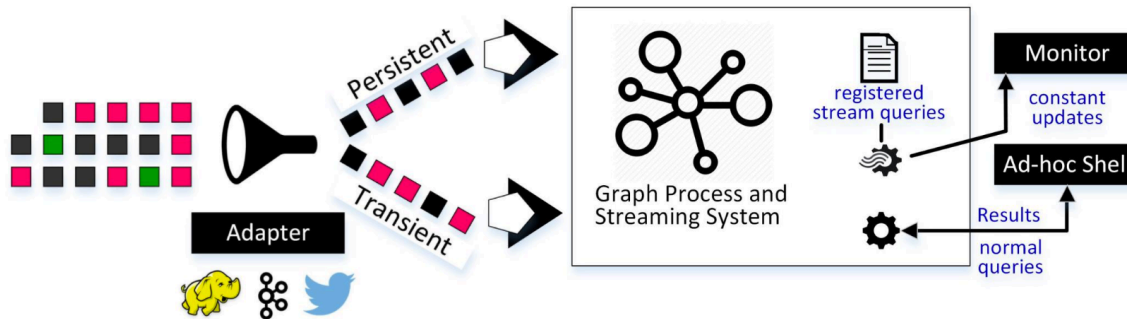


► **low-latency, concurrent** queries over large datasets

- A 6-node cluster w/ **RDMA**
- LUBM-10240 (**1.4B** Triples)
- Up to **185 K queries/sec** with **0.80 msec** (geo-mean) median latency
- 180-740X throughput increase over Trinity.RDF/TriAD

Wukong-S: Streaming Processing

1. SQL-like API for graph query over streams
 2. Decoupled design of RDF Store for efficiently combining streams and persistent data
 3. Native strong consistency guarantee
- ▶ **low-latency, concurrent** streaming queries over large datasets
- 148K Queries/s for a 6-node cluster w/ **RDMA**
 - 1.38ms medium latency for CityBench (541ms for Spark streaming)



Rethink Systems Software

Exploiting Parallelism is Hard!

Quote from Dijkstra

I am convinced more than ever that **this type of work** is very difficult, and that every effort to do it with other than **the best people** is doomed to either **failure** or **moderate success at enormous cost**.

-- Edsger Dijkstra

“The Structure of the ‘THE’ “, Multiprogramming System 1968

Quote from Thacker

In the era of many-core systems, programs can't be written by only “the best people”.

Chuck Thacker

“Improving the future by examining the past”
Turing Lecture Series, 2010.

Ideal Multicore Scalability



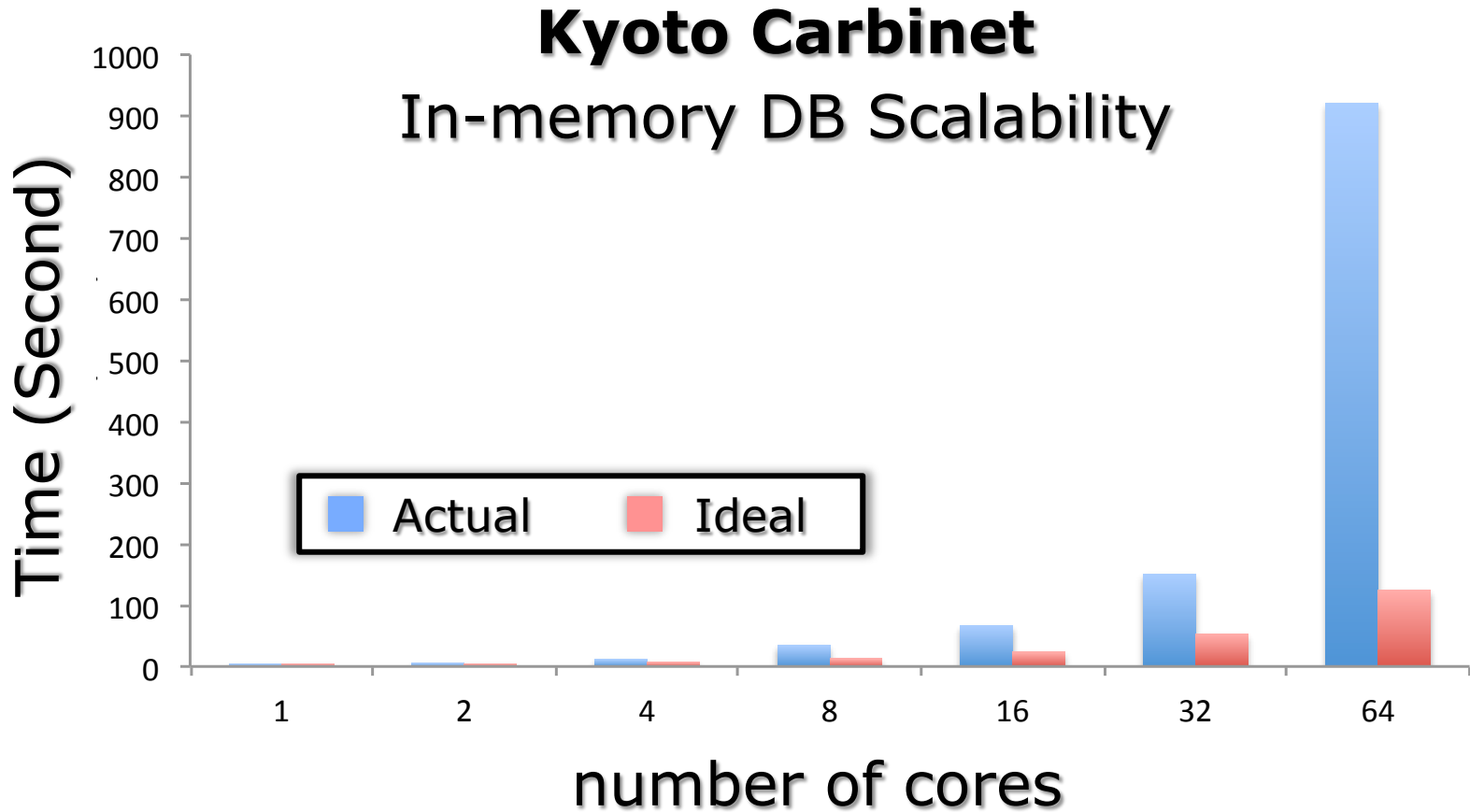
Credit:
Erlang@Sina
Weibo

Multicore Scalability in Reality



Credit:
Erlang@Sina
Weibo

Database Scalability Issue



8 socket * 8 cores, AMD

Sync Constructs Matter

Sync constructs meet multicore

- Parallelism: need to unleash more parallelism
- Critical section efficiency: reduce cache traffics

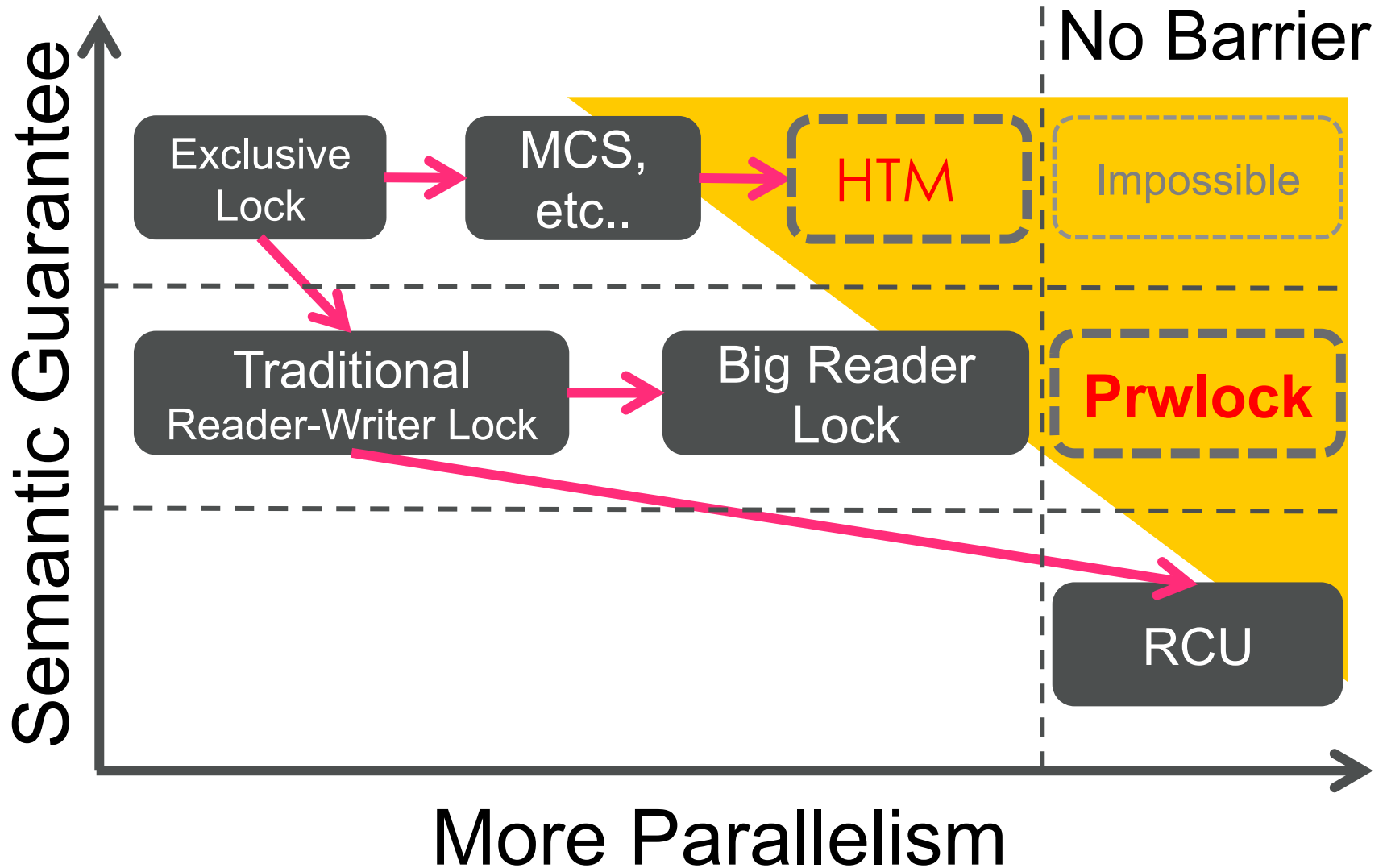
One small atomic instruction can **collapse** whole application performance for many-cores

- Kaashoek, APSys'12 Keynote

Insights from prior work



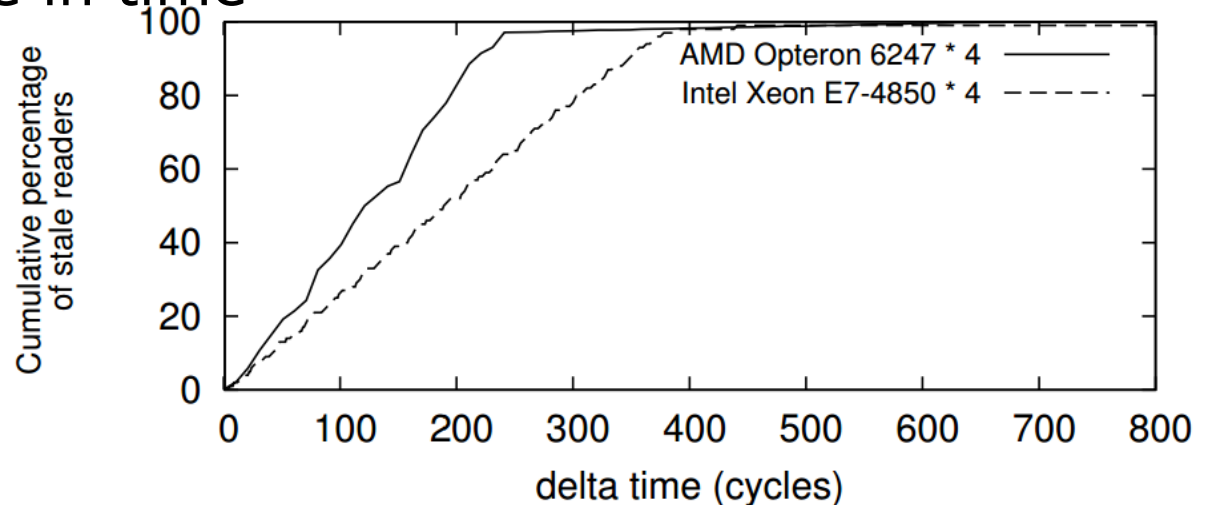
Synchronization Evolution



Bounded Staleness: Hardware's Habit

Shared memory write becomes
globally visible in a short time

- Most memory write are visible to others within 400 cycles without memory barrier
- Memory barrier is not necessary to observe **newest** state in time



Passive Reader-Writer Lock (Usenix ATC 2014)

Principle: **common case fast, rare case correct**

No memory barrier in common case

→ Leverage **bounded staleness** to wait until a reader see a writer's version

Bounded lock acquisition latency through IPIs

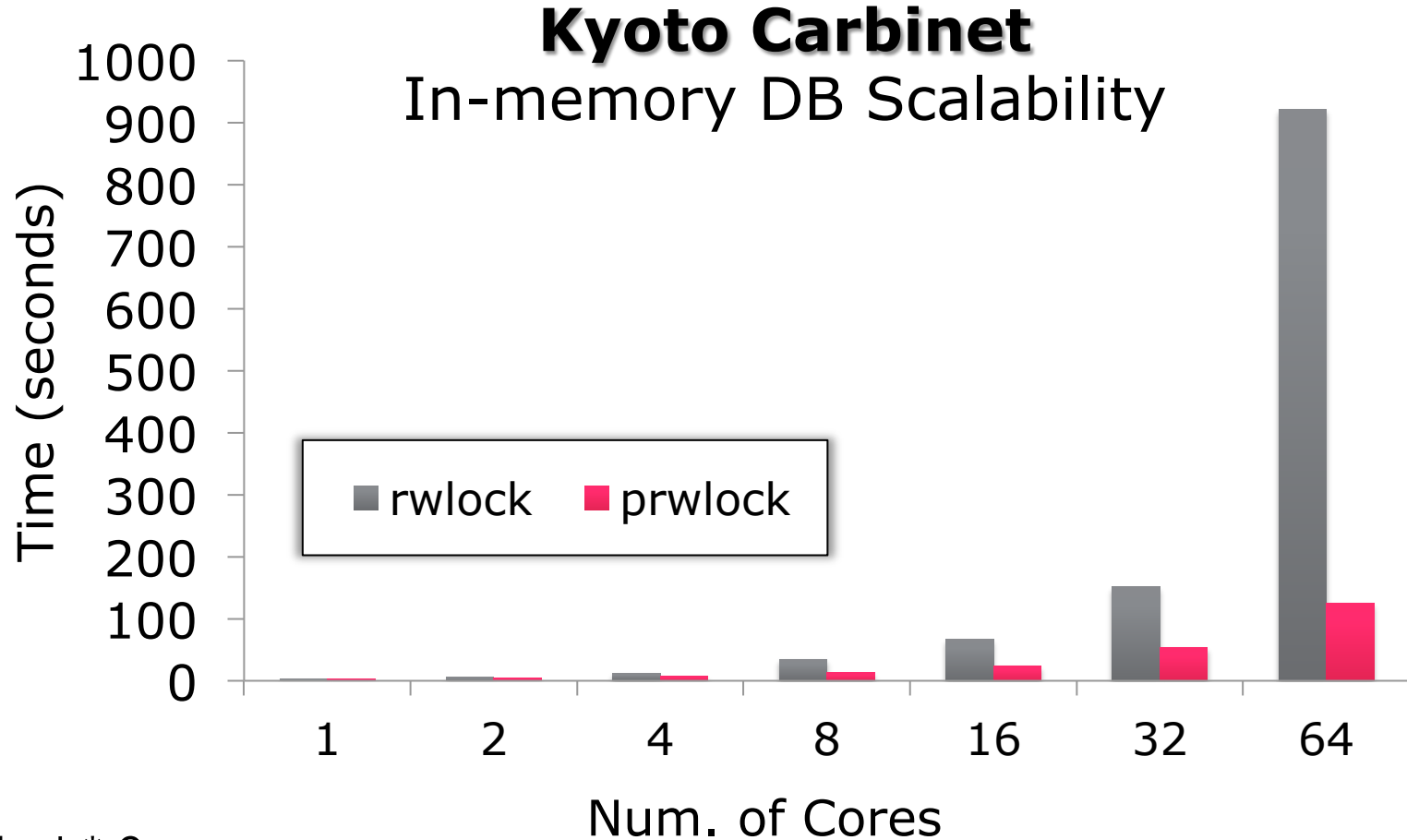
→ Voluntarily sending IPIs to **straggling** readers to query its status

Results

→ Similar performance characteristics with RCU

→ Same semantic guarantee with rwlock

Performance on In-memory DB



8 socket * 8 cores,
AMD Operon

Scalable Consensus for Read-copy Update (TPDS 2016)

Read-copy update is widely used for kernel sync

- Readers require no memory barrier
- Concurrent execution of readers and a single writer
- Reclaimer detect if an object is safe to be reclaimed
 - Usually requires at least a scheduler tick

Fast reclamation with fast consensus

- Use versions to detect liveness
- No memory barrier in readers
- Very fast on common case

Result

- Faster consensus for RCU
- Better update performance

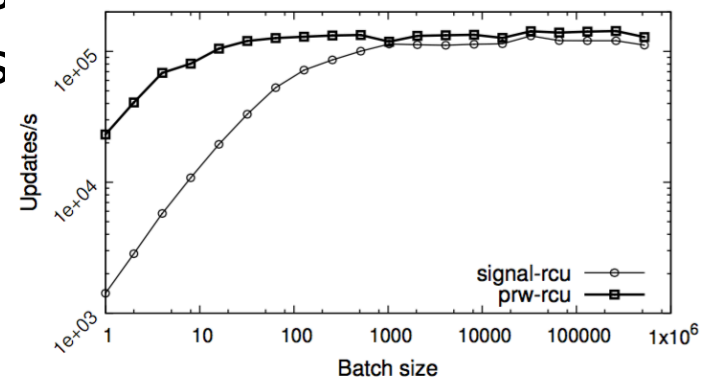


Fig. 22: Update performance with batch size

Fence-free Synchronization (TPDS 2017)

Fence causes high cost

- Serialize processor pipelines
- Drain store buffers
- Existing fence are pessimistic: overall constrained

Sync-order: fence-free synchronization

- Detect/prevent dangerous inter-processor dependencies
- Using sync-Vars to reducing Detection Overhead

Result

- Eliminate almost all unnecessary stalls
- Better multicore performance

Architecture Support for IMC (IEEE CAL 2015)

Example: what a transaction needs?

ACID: Atomicity, Consistency, Isolation and
Durability

What current hardware provides?

Transactional memory: ACI, missing “durability”

Data loss/inconsistency during a machine crash

Persistent transactional memory

Adding persistency support for TM to support ACID

Combining NVM with TM

Simplify the writing of transaction code

Summary

In-memory transactions demands **high throughput** and **low latency**

RDMA: helps bridge the gap from incommensurate scaling for in-memory transactions

Achieving **orders-of-magnitude** lower **latency** & higher **throughput** than prior state-of-the-art centralized and distributed systems

Thanks



Questions?



Institute of Parallel and Distributed Systems
(IPADS)

<http://ipads.se.sjtu.edu.cn>



Backup

Comparison with FaRM

DrTM-OCC follows the distributed OCC scheme of FaRM

