Low Overhead Concurrency Control for Partitioned Main Memory Databases

Evan P. C. Jones    Daniel J. Abadi    Samuel Madden
Banks
Payment Processing
Airline Reservations
E-Commerce
Web 2.0
Problem:

Millions of transactions per second
Problem:

Millions of transactions per second
Problem:

Millions of transactions per second

= $$$
Alternative: H-Store Project

Redesign specifically for OLTP

Prototype: ~10X throughput

Idea: Remove un-needed features

H-Store: High Throughput OLTP

Redesign DB specifically for OLTP
Prototype: ~10X throughput
Main memory database

Concurrency control consumes
~30-40% of CPU time
CPU Cycle Breakdown for Shore on TPC-C New Order
Source: Harizopoulos, Abadi, Madden and Stonebraker, “OLTP Under the Looking Glass”, SIGMOD 2008
CPU Cycle Breakdown for Shore on TPC-C New Order

Source: Harizopoulos, Abadi, Madden and Stonebraker, “OLTP Under the Looking Glass”, SIGMOD 2008
Speculative Concurrency Control

Eliminate fine-grained access tracking (locks or read/write sets)
Eliminate undo logs (where possible)

Up to 2X faster than locking for appropriate workloads
Why Support Concurrency?

Use idle resources:
- disk stalls
- user stalls
- main memory
- stored procedures

Physical resources:
- multiple CPUs
- multiple disks
- partition per core

Long running txns:
- don’t do them
H-Store: Single thread engine

Assumptions:

Database divided into partitions
Transactions access one partition (mostly)
Mapping procedures to partitions is given
Total data fits in memory of N machines
Partitions are replicated on 2 machines
System Overview
Single Partition Transaction

Client

Primary

Backup

1
Single Partition Transaction

Client → Primary → Backup

1

2
Single Partition Transaction

Client

Primary

Backup

1

execute

2
Single Partition Transaction

Client

Primary

execute

Backup
Single Partition Transaction

Client

Primary

execute

Backup
Single Partition Transaction

Client

Primary

Backup

execute execute
Single Partition Transaction

Clients

H-Store

Client Library

Client Library

Client Library

Coordinator

Partition 1

Primary

Partition 2

Primary

Partition 1

Backup

Partition 2

Backup
Single Partition Transaction

Clients

H-Store

Client Library

Partition 1
Primary

(formatter) 1

Coordinator

Partition 1
Backup

(formatter) 2

Client Library

Partition 2
Primary

Client Library

Partition 2
Backup
Single Partition Transaction

Clients

H-Store

Client Library

Partition 1

Primary

Backup

Coordinator

Client Library

Partition 2

Primary

Backup

Client Library

Partition 2
Single Partition Transaction

Clients

H-Store

1. Clients

2. H-Store

3. Single Partition Transaction

4. Coordinator

Partition 1

Primary

Backup

Partition 2

Primary

Backup

Partition 2
Single Partition Transaction
Single Partition Transaction

Clients

H-Store

Partition 1

1

Partition 1

2

Partition 1

3

Partition 2

4

Partition 2

Primary

Backup

Primary

Backup

Coordinator
Single Partition Transaction

Clients

H-Store

Partition 1

Primary

Backup

Partition 2

Primary

Backup

Coordinator
Single Partition Transaction
Single Partition Transaction
Single Partition Transaction

Clients

H-Store

Partition 1

Primary

Backup

Partition 2

Primary

Backup

Partition 1

Coordinator

Partition 2

Client Library

Client Library

Client Library
Single Partition Transaction

Clients

H-Store

Partition 1

Primary

Backup

Partition 2

Primary

Backup

Coordinator

Client Library

1

4

2

3
Not Perfectly Partitionable?

Example: users and groups

Many applications are *mostly* partitionable

e.g. TPC-C: 11% multi-partition transactions
Distributed Transactions

Need two-phase commit (consensus)

Simple solution:

**block** until the transaction finishes

Introduces network stall (**bad**)
Blocking Multi-Partition

Clients

H-Store

1

Coordinator

Partition 1  Primary

Partition 1  Backup

Partition 2  Primary

Partition 2  Backup
Blocking Multi-Partition
Blocking Multi-Partition

[Diagram showing a system with Clients, H-Store, Coordinator, and Partitions 1 and 2. Clients connect to Client Libraries which in turn connect to the Coordinator. Partitions 1 and 2 are labeled as Primary and Backup respectively.]
Blocking Multi-Partition

1

Coordinator

2

P1 Primary

P1 Backup
Blocking Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup
Blocking Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

1

2

3

execute
Blocking Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

execute

1
2
3
4
Blocking Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

execute
Blocking Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

1

2

3

execute

4

5
Blocking Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

execute
Blocking Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

execute

execute
Blocking Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

execute

execute

network stall
Blocking Multi-Partition

Clients

H-Store

Client Library

Client Library

Client Library

Coordinator

Partition 1

Primary

Partition 1

Backup

Partition 2

Primary

Partition 2

Backup
Blocking Multi-Partition

Clients

H-Store

Client Library

Coordinator

Partition 1

Primary

Partition 2

Primary

Backup

Partition 1

Partition 2

Backup
Blocking Multi-Partition

Clients

H-Store

Client Library

Partition 1
Primary

Partition 1
Backup

Partition 2
Primary

Partition 2
Backup

Coordinator

1

2

2

3

4

5

5

3

4
Blocking Multi-Partition

Clients
H-Store

Client Library

Partition 1
Primary
Backup

Partition 2
Primary
Backup

Coordinator

1 2 3 4 5 6
Two-Phase Locking

+ Execute non-conflicting txns during stall
+ No need to order in advance
  – Locking overhead
  – Deadlocks

**Optimization**: turn off locks and undo logging when no multi-partition transactions
Speculative CC

While waiting for commit/abort, speculatively execute other transactions

+ No locks; no read/write sets
  – Need global transaction order
  – Cascading aborts
Speculative Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

1 2 3 4 5

execute execute
Speculative Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

1

2

3

4

5

execute

execute
Speculative Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

execute

execute
Speculative Multi-Partition

Client

Coordinator

P1 Primary

P1 Backup

1 → 2 → 3 → execute

2 → 4 → 5 → execute

3 → 6

4 → 6
Speculative Multi-Partition
Speculation Limitation

Transactions with multiple “rounds” of work: need network stall

Example:
1. Read $x$ on partition 1, $y$ on partition 2
2. Update $x = f(x, y); y = f(x, y)$
Speculative Multi-Partition

Clients

H-Store

Client Library

Client Library

Client Library

Primary

Primary

Backup

Backup

Partition 1

Partition 2

Partition 1

Partition 2

Coordinator

1

2

2

3

3
Speculative Multi-Partition
Speculative Multi-Partition

Clients

H-Store

Client Library

Client Library

Client Library

Coordinator

Partition 1

Primary

Backup

Partition 1

Partition 2

Primary

Backup

Partition 2
Speculative Multi-Partition

Clients

H-Store

Coordinator

Partition 1
Primary
Backup

Partition 2
Primary
Backup
Speculative Multi-Partition

Clients

H-Store

Partition 1

Primary

Backup

Partition 2

Primary

Backup
Speculative Multi-Partition

Clients

H-Store

Client Library

Client Library

Client Library

Coordinator

Partition 1

Partition 2

Primary

Backup

Partition 1

Partition 2

Backup

Primary
Speculative Multi-Partition

Clients

H-Store

Client Library

Client Library

Client Library

Coordinator

Partition 1

Partition 1

Partition 2

Partition 2

Primary

Backup

Primary

Backup
Speculative Multi-Partition

Clients

H-Store

Client Library

Partition 1

Primary

Backup

Partition 2

Coordinator

Partition 1

Partition 2

Backup
Microbenchmark

Two partitions of a single table

\[(id \text{ INTEGER PRIMARY KEY, value \text{ INTEGER}})\]
Microbenchmark

Single partition transaction:
read/write keys on one partition

Multi-partition transaction:
access half keys from each partition

single partition work = multi-partition work

No deadlocks, no aborts, no conflicts
TPC-C Based

~11% multi-partition transactions
More complex locking
Many conflicts
Some deadlocks
Some aborts
The graph shows the number of transactions per second as a function of the number of warehouses. There are three lines representing different scenarios:

- **Speculation** (solid black line)
- **Blocking** (dashed red line)
- **Locking** (dotted green line)

As the number of warehouses increases, the number of transactions per second for all scenarios generally decreases. However, the rate of decrease is different for each scenario. The Speculation line shows a more gradual decrease compared to the Blocking and Locking lines, which have sharper declines at the beginning but stabilize at higher transaction rates. The Locking line shows the least change over the range of warehouses shown in the graph.
Speculative CC

better for “mostly partitionable” apps on main memory DBs

Up to 2X throughput
No locking overhead
No deadlocks
Transactions/second vs. Multi-Partition Transactions

- Speculation
- Local Speculation
- Blocking
- Locking
- Optimistic
Transactions/second vs. Warehouses

- Speculation
- Blocking
- Locking
- Optimistic