Low Overhead Concurrency Control for Partitioned Main Memory Databases

Evan Jones, Daniel Abadi, Sam Madden

evanj@mit.edu

Problem: How do we make OLTP workloads go 10x faster?

Solution: Single thread execution; Partitioning; Avoid locks; Use speculation

Traditional Concurrency

Idle Resources:

- Wait for disk
- Wait for user

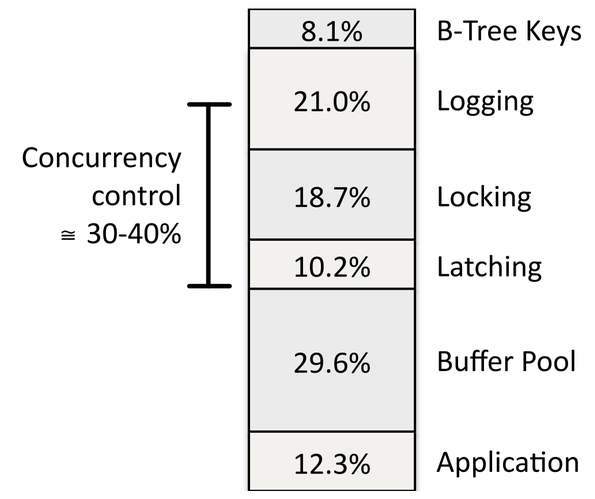
Physical Concurrency:

- Multiple CPUs, disks

Our Approach: H-Store

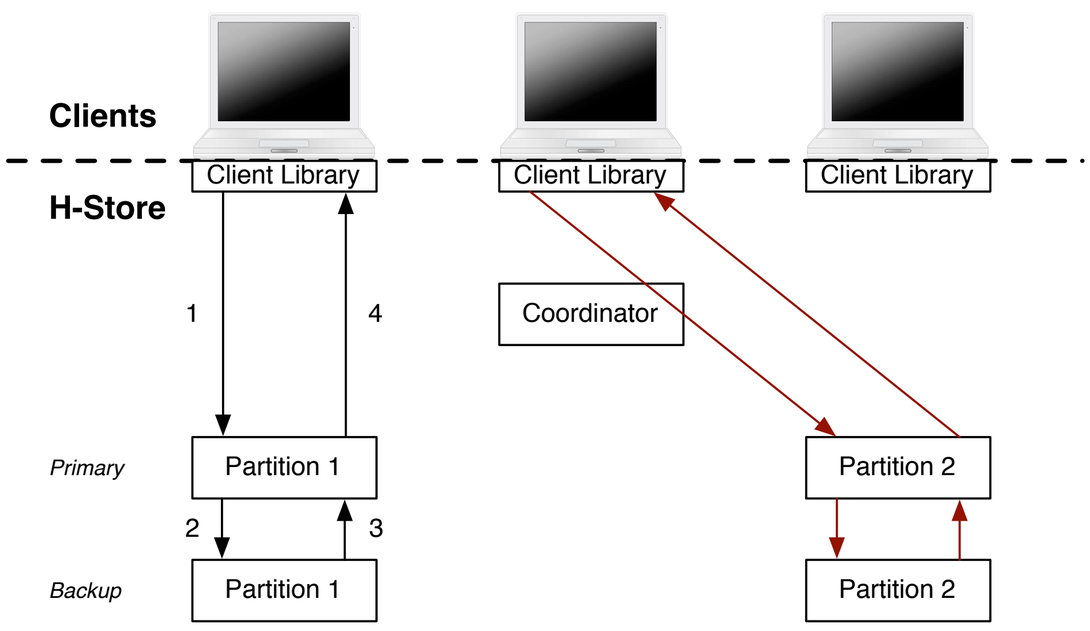
- Main memory
- Stored procedures
- Multiple partitions

% CPU Cycles (Shore)



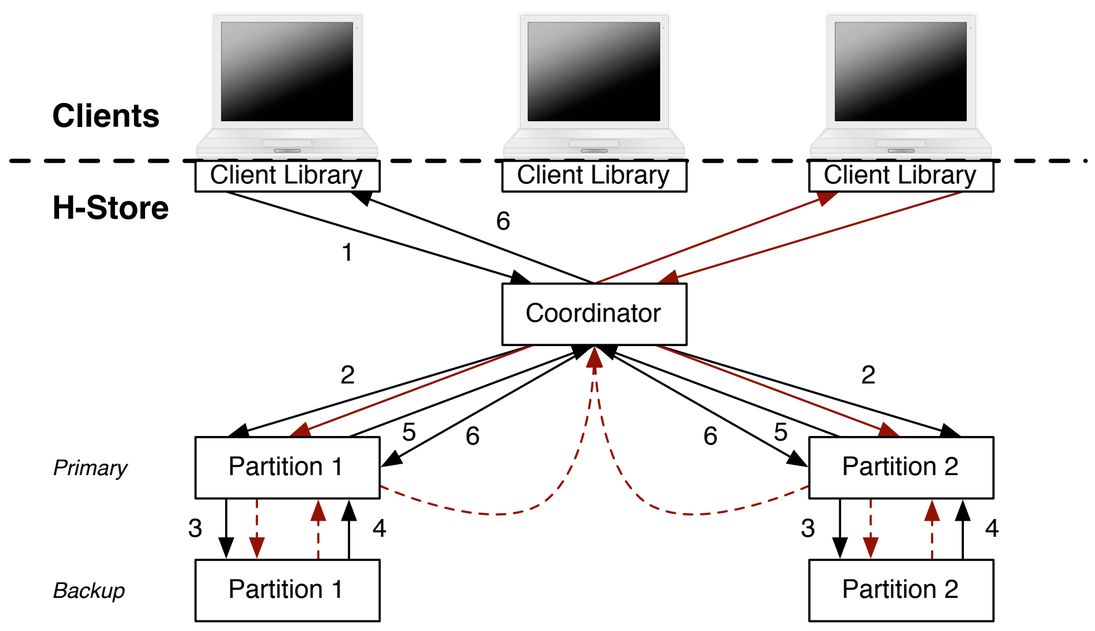
Single Partition Transactions

No locks, no undo logging: no overhead



Multi-Partition Transactions

Two-phase commit; **network stall (bad)**



Low Overhead Concurrency Control: Do useful work during network stall

Speculation: Speculate next transactions during stall, after txn is prepared

- Best for simple multi-partition transactions: one round of work on partitions

Locking: Don't acquire locks if only executing single partition transactions

- Best for workloads with complex transactions; inter-partition communication

Experimental Results

Microbenchmark: Two partitions; Change fraction of multi-partition transactions

TPC-C like: Two partitions varying the number of warehouses