Revisiting OLTP Databases

Today's Plan

- Quick Review of Traditional OLTP Systems
- Major Hardware Changes
- OLTP Through the Looking Glass
- New Systems and Ideas
  - Multi-threading
  - Partitioning vs no-partitioning
  - Indexes
  - Anti-caching
- Other Issues with In-Memory Databases
  - Analytics?
  - Next steps?

Review of OLTP

- Traditional focus on avoiding disk I/Os
- Some key things:
  - Buffer Manager
    - Steal and No force Policies; Pinning/Unpinning
  - Concurrency Control
    - Locking or Optimistic (more in-depth treatment later)
  - Recovery and Durability
    - Write-ahead Logging
    - Logging also used for replication (log shipping)
  - B+-Tree Indexes
    - Require own locking protocols
  - Stored procedures always supported, but not treated differently

Major Hardware Changes

- Huge memories → disk I/O much less important
  - Also: bottlenecks shift
  - E.g., locking becomes relatively very expensive
  - Things like: pointer chasing, predicate evaluations, data copying, network I/O, all become crucial
- Multi-core
  - Much more parallelism than before → latching contention becomes critical
- Cache misses quite expensive
  - Memory-optimized indexes maybe worse than B+-Trees
- Using mmap can help solve some problems, but not all

OLTP through the Looking Glass

- Where does the time go?

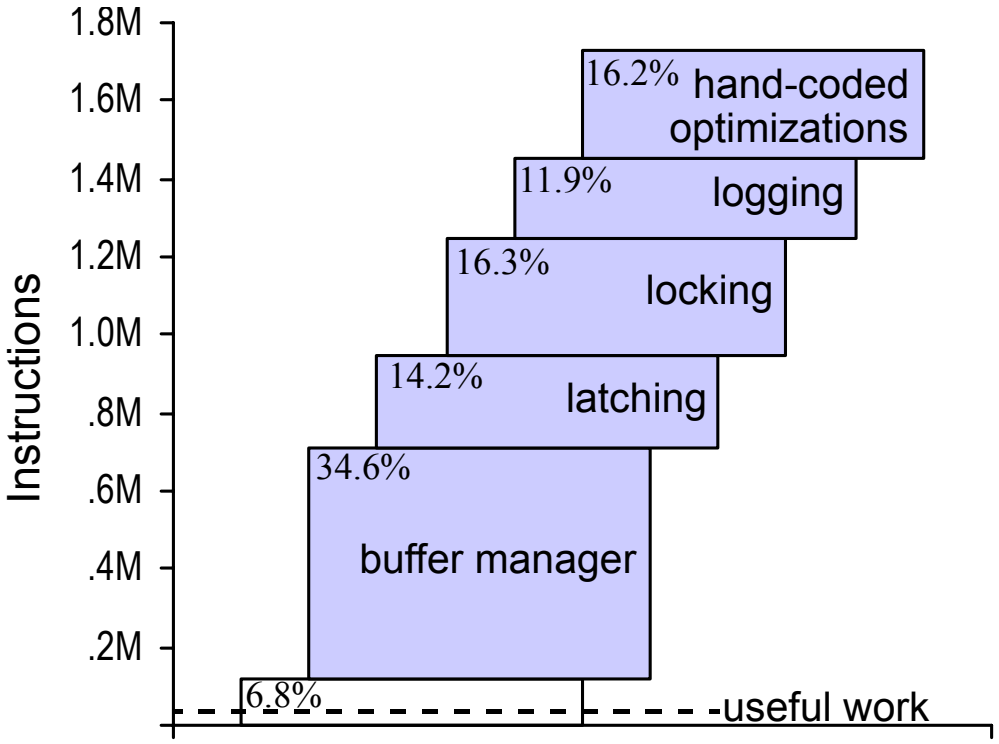

Figure 1. Breakdown of instruction count for various DBMS components for the New Order transaction from TPC-C. The top of the bar-graph is the original Shore performance with a main memory resident database and no thread contention. The bottom dashed line is the useful work, measured by executing the transaction on a no-overhead kernel.

Work Since Then

- Many new systems shifting towards in-memory solutions
  - Hekaton, H-Store/VoltDB, SAP HANA, …
- Some issues:
  - Multi-threading and contention
    - Need to rethink latching altogether
    - Prefer to use lock-free data structures if possible
  - Partitioning vs no-partitioning
    - Kinda depends on the workload
    - Lot of papers on the topic

Work Since Then

- Some issues:
  - Weaker notions of consistency
    - Eventual consistency is too weak, most people moving away from it
    - Snapshot Isolation/Read Committed give more clear guarantees
  - Indexes
  - Anti-caching
  - Command logging
    - i.e., log transactions rather than the specific updates
    - Only works if there is deterministic execution
- Other Issues with In-Memory Databases
  - Analytics?
  - Next steps?