Evolving the architecture of a DBMS for modern hardware

Paul Larson
Principal researcher
Microsoft Research
Time travel back to circa 1980

Typical machine was VAX 11/780
- 1 MIPS CPU with 1KB of cache memory
- 8 MB memory (maximum)
- 80 MB disk drives, 1 MB/second transfer rate
- $250K purchase price!

Basic DBMS architecture established
- Rows, pages, B-trees, buffer pools, lock manager, ....

Still using the same basic architecture!
But hardware has evolved dramatically

**US$ per GB of PC class memory**
Source: www.jcmit.com/memoryprice.htm

**No of cores/socket over time**

**Shrinking memory prices**

**More and more cores**
How to evolve SQL Server’s architecture?

Apollo column store
Column store technology integrated into SQL Server
Targeted for data warehousing workloads
First installment in SQL 2012, enhancements in SQL 2014

Hekaton main-memory engine
Main-memory database engine integrated into SQL Server
Targeted for OLTP workloads
Will ship in SQL 2014
Hekaton architectural pillars

Main-Memory Optimized
- Optimized for in-memory data
- Indexes (hash, range) exist only in memory
- No buffer pool
- Stream-based storage (log and checkpoints)

Designed for High Concurrency
- Multi-version optimistic concurrency control with full ACID support
- Core engine using lock-free algorithms
- No lock manager, latches or spinlocks

T-SQL Compiled to Machine Code
- T-SQL compiled to machine code via C code generator and VC
- Invoking a procedure is just a DLL entry-point
- Aggressive optimizations @ compile-time

Integrated into SQL Server
- Integrated queries & transactions
- Integrated HA and backup/restore
- Familiar manageability and development experience

Steadily declining memory price
Many-core processors
Stalling CPU clock rate
Total Cost of Ownership

Hardware trends
Business Driver

Steadily declining memory price
Many-core processors
Stalling CPU clock rate
Total Cost of Ownership
Non-blocking execution

Goal: highly concurrent execution, full CPU utilization
No thread switching, waiting, or spinning during execution of a transaction

Lead to three design choices
Use only latch-free data structure
Multi-version optimistic concurrency control
Allow certain speculative reads (with commit dependencies)

Result: great majority of transactions run up to final log write without ever blocking or waiting
Rows are multi-versioned
Each row version has a valid time range indicated by two timestamps
A version is visible if transaction read time falls within version’s valid time
Why MV optimistic concurrency control?

Readers don’t block writers and vice versa
No lock manager, no deadlocks
Highly parallel
A single synchronization point: get transaction end timestamp

Lower isolation level => less work
Snapshot Isolation: no validation, minimal overhead

Performs well even under high contention
Handles long read-only transaction well
Scalability under extreme contention
(1000 row table, core Hekaton engine only)

**Work load**
80% read-only txns (10 reads/txn)
20% update txns (10 reads + 2 writes/txn)

** Serializable isolation level**

**Processor:** 2 sockets, 12 cores

**Standard locking but optimized for main memory**

1V/L throughput limited by lock thrashing
Hekaton components and SQL integration
Query and transaction interop

Regular SQL queries can access Hekaton tables like any other table
Slower than through a compiled stored procedure
A query can mix Hekaton tables and SQL tables
A transaction can update both types of tables
Throughput under high contention

Throughput improvements
Converting table but using interop: 3.3X higher throughput
Converting table and stored procedure: 15.7X higher throughput

Workload:
- read/insert into a table with a unique index
- append a batch of 100 rows
- read last inserted batch of rows
Initial customer experiences

**Bwin** – large online betting company
Application: HTTPS session state
Current max throughput: 15,000 requests/sec
Throughput with Hekaton: 250,000 requests/sec

**EdgeNet** – provides up-to-date inventory information
Application: rapid ingestion of inventory data from retailers
Current max ingestion rate: 7,450 rows/sec
Hekaton ingestion rate: 126,665 rows/sec
Enables moving to continuous, online ingestion from once-a-day batch ingestion