OldSQL vs. NoSQL vs. NewSQL on New OLTP

Michael Stonebraker, CTO
VoltDB, Inc.
Old OLTP

- Remember how we used to buy airplane tickets in the 1980s
  - By telephone
  - Through an intermediary (professional terminal operator)

- Commerce at the speed of the intermediary

- In 1985, 1,000 transactions per second was considered an incredible stretch goal!!!!
  - HPTS (1985)
How has OLTP Changed in 25 Years?

The internet

+ Client is no longer a professional terminal operator
+ Instead Aunt Martha is using the web herself
+ Sends volume through the roof
How has OLTP Changed in 25 Years?

PDAs and sensors

+ Your cell phone is a transaction originator
+ Everything is being geo-positioned by sensors (marathon runners, your car, ....)
+ Sends volume through the roof
How has OLTP Changed in 25 Years?

The definitions

+ “Online” no longer exclusively means a human operator
  - The oncoming data tsunami is often device and system-generated
+ “Transaction” now transcends the traditional business transaction
  - High-throughput ACID write operations are a new requirement
+ “HA” and “durability” are now core database requirements
Examples

Maintain the state of multi-player internet games
Real time ad placement
Fraud/intrusion detection
Risk management on Wall Street
New OLTP Challenges

You need to **ingest** the firehose in real time

You need to **process, validate, enrich** and **respond** in real-time

You often need **real-time** analytics
Solution Choices

- **OldSQL**
  - + Legacy RDBMS vendors

- **NoSQL**
  - + Give up SQL and ACID for performance

- **NewSQL**
  - + Preserve SQL and ACID
  - + Get performance from a new architecture
OldSQL

Traditional SQL vendors (the “elephants”)

+ Code lines dating from the 1980’s
+ “bloatware”
+ Not very good at anything
  — Can be beaten by at least an order of magnitude in every vertical market I know of
+ Mediocre performance on New OLTP
  — At low velocity it doesn’t matter
  — Otherwise you get to tear your hair out
DBMS Landscape – Performance Needs
One Size Does Not Fit All -- Pictorially

Elephants only get “the crevices”
Reality Check

- TPC-C CPU cycles
- On the Shore DBMS prototype
- Elephants should be similar
The Elephants

- Are slow because they spend all of their time on overhead!!!
  + Not on useful work

- Would have to re-architect their legacy code to do better
To Go a Lot Faster You Have to......

- Focus on overhead
  - Better B-trees affects only 4% of the path length

- Get rid of ALL major sources of overhead
  - Main memory deployment – gets rid of buffer pool
    - Leaving other 75% of overhead intact
    - i.e. win is 25%
Long Term Elephant Outlook

- Up against “The Innovators Dilemma”
  - Steam shovel example
  - Disk drive example
  - See the book by Clayton Christenson for more details

- Long term drift into the sunset
  - The most likely scenario
  - Unless they can solve the dilemma
NoSQL

- Give up SQL
- Give up ACID
Give Up SQL?

- Compiler translates SQL at compile time into a sequence of low level operations
- Similar to what the NoSQL products make you program in your application
- 30 years of RDBMS experience
  - Hard to beat the compiler
  - High level languages are good (data independence, less code, ...)
  - Stored procedures are good!
    - One round trip from app to DBMS rather than one one round trip per record
    - Move the code to the data, not the other way around
Give Up ACID

- If you need data accuracy, giving up ACID is a decision to tear your hair out by doing database “heavy lifting” in user code
- Can you guarantee you won’t need ACID tomorrow?

ACID = goodness, in spite of what these guys say
Who Needs ACID?

- Funds transfer
  - Or anybody moving something from X to Y

- Anybody with integrity constraints
  - Back out if fails
  - Anybody for whom “usually ships in 24 hours” is not an acceptable outcome

- Anybody with a multi-record state
  - E.g. move and shoot
Who needs ACID in replication

- Anybody with non-commutative updates
  + For example, + and * don’t commute

- Anybody with integrity constraints
  + Can’t sell the last item twice....

- Eventual consistency means “creates garbage”
NoSQL Summary

- Appropriate for non-transactional systems
- Appropriate for single record transactions that are commutative
- Not a good fit for New OLTP
- Use the right tool for the job

Interesting ...

Two recently-proposed NoSQL language standards – CQL and UnQL – are amazingly similar to (you guessed it!) SQL
NewSQL

- SQL
- ACID
- Performance and scalability through modern innovative software architecture
NewSQL

- Needs something other than traditional record level locking (1st big source of overhead)
  - timestamp order
  - MVCC
  - Your good idea goes here
NewSQL

- Needs a solution to buffer pool overhead (2\textsuperscript{nd} big source of overhead)
  - Main memory (at least for data that is not cold)
  - Some other way to reduce buffer pool cost
NewSQL

- Needs a solution to latching for shared data structures (3rd big source of overhead)
  + Some innovative use of B-trees
  + Single-threading
  + Your good idea goes here
Needs a solution to write-ahead logging (4th big source of overhead)
  + Obvious answer is built-in replication and failover
  + New OLTP views this as a requirement anyway

Some details
  + On-line failover?
  + On-line failback?
  + LAN network partitioning?
  + WAN network partitioning?
A NewSQL Example – VoltDB

- Main-memory storage
- Single threaded, run Xacts to completion
  - No locking
  - No latching
- Built-in HA and durability
  - No log (in the traditional sense)
Yabut: What About Multicore?

- For a K-core CPU, divide memory into K (non-overlapping) buckets
- i.e. convert multi-core to K single cores
Where all the time goes... revisited

Before

- Latching 24%
- Locking 24%
- Buffer Pool 24%
- Recovery 24%
- Useful Work 4%

VoltDB

- Useful Work 95%
- Locking 5%
Current VoltDB Status

- Runs a subset of SQL (which is getting larger)
- On VoltDB clusters (in memory on commodity gear)
- No WAN support yet
  + Working on it right now
- 50X a popular OldSQL DBMS on TPC-C
- 5-7X Cassandra on VoltDB K-V layer
- Scales to 384 cores (biggest iron we could get our hands on)
- Clearly note this is an open source system!
Summary

Old OLTP

- OldSQL for New OLTP
  - Too slow
  - Does not scale

- NoSQL for New OLTP
  - Lacks consistency guarantees
  - Low-level interface

New OLTP

- NewSQL for New OLTP
  - Fast, scalable and consistent
  - Supports SQL