

# NewSQL

## (vs NoSQL or OldSQL)

Michael Stonebraker, CTO  
VoltDB, Inc.

# How Has OLTP Changed in 25 years

- Professional terminal operator has been dis-intermediated (by the web)
  - + Sends volume through the roof
- Transactions originate from PDAs
  - + Sends volume through the roof

# How Has OLTP Changed in 25 years

- Most OLTP can fit in main memory
  - + 1 Terabyte is a reasonably big OLTP data base
  - + And fits in a modest 32 node cluster with 32 gigs/node
- Nobody will send a message to a user inside a transaction
  - + Aunt Martha may have gone to lunch

# How Has OLTP Changed in 25 years

- In 1985, 1,000 transactions per second was considered an incredible stretch goal!!!!
  - + HPTS (1985)
- Now the goal is 2 – 4 orders of magnitude higher

# New OLTP

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

You need to perform high volume OLTP

You often need **real-time** analytics



# Solution Options

OldSQL (the RDBMS elephants)

NoSQL (the 75 or so companies that suggest abandoning both SQL and ACID)

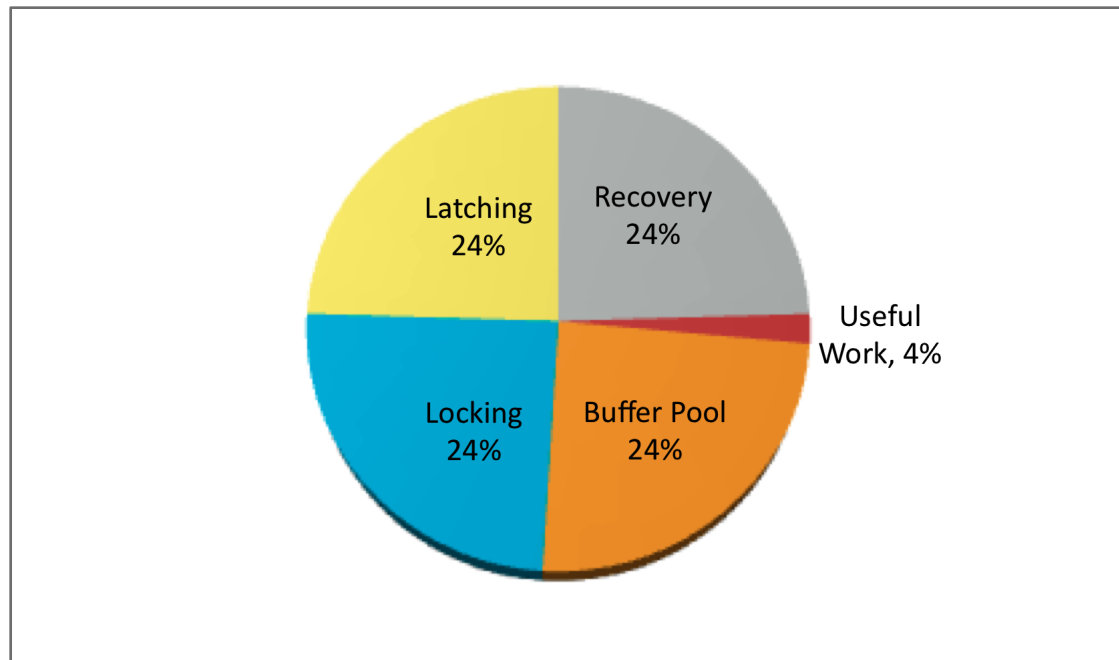
NewSQL (the companies that keep SQL and ACID, but with a different architecture than the elephants)

# The Elephants (Unless You Squint)

- Disk-based
- Drank the Mohan koolaid (Aries)
- Listened to Mike Carey (dynamic record-level locking)
- Active-passive replication
- Multi-threaded

# 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 completely re-architect their legacy code to do better
- Non-starter on New OLTP

# NoSQL

- Give up SQL

- + Interesting to note that Cassandra and Mongo are moving to (yup) SQL

- Give up ACID

- + If you need ACID, this is a decision to tear your hair out by doing it in user code

- + Can you guarantee you won't need ACID tomorrow?



# What's More.....

- Much of the Old OLTP overhead is still there
  - + Multi-threaded
  - + Disk-based
  - + Lighter weight locking
  - + Some have WALs

# 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%

# NewSQL

- Keep SQL
- Keep ACID
- Go fast using some non-traditional architecture

Examples: VoltDB, NuoDB (NimbusDB), Clustrix, Akiban

# NewSQL Example -- VoltDB

- Shared nothing architecture
  - + Sharded
  - + Hash or range partitioning
- Main memory DBMS
- Single threaded
  - + Main memory divided (statically right now) per core
  - + No shared data structures
- Open source
  - + Cheaper sales model (no 4-legged sales teams)

# NewSQL Example -- VoltDB

- Stored procedure interface
  - + OLTP updates clearly use this model
  - + Compiled on the fly for ad-hoc queries
- All tables replicated (K-safety)
  - + Identically sharded
  - + LAN for now – we are just starting to think about WAN replication
  - + Active-active replication with failover/failback (HA built-in)
  - + ACID!!!!

# NewSQL Example: VoltDB

- Single node transactions
  - + Run to completion at the correct node (for each replica)
  - + In timestamp order (no locking)
- Multi-node transactions
  - + Moving to speculative execution – with backout if necessary
  - + There are a bunch of other ideas that we might try also

# VoltDb Summary

- No buffer pool
- No locking (no Mike Carey)
- No WAL (no Mohan)
- No threading overhead

# Fast!!!!!!

- 60 X an elephant on TPC-C
- 8 X Cassandra
- 1 X Memcached (using key-value stored procedure layer)

# Scalable!!!

- Linearly with the number of cores in a node
  - + No shared data structures
- Linearly with the number of nodes
- Up to the biggest cluster (30 X 8) we could get our hands on
  - + Ran 3.4M simple xacts/sec.

# What About Cluster-wide Failures?

- How?
  - + Power failure with no UPS
  - + Bohr bugs
  - + User errors (RM \*)
- Can't cause a data loss.....

# What Does VoltDB Do?

- Transaction consistent checkpointing (currently continuous & asynchronous)
  - + cheap
- Stored procedure log (#4, “diet coke”)
  - + Group commit

# What Does VoltDB Do?

- Restore latest checkpoint & re-execute stored procedures after checkpoint
- Compared this to Aries (paper almost ready)
  - + Run-time overhead way way way smaller
  - + Recovery time longer, since we don't currently checkpoint indexes
  - + With indexes, it would be about the same.....
- Summary: Mohan lite.....

# VoltOne – For the Low End

- Single node system
- With command logging
- No replication
- No HA

# What if your DB is Too Big?

- If OLTP data gets stale
  - + Age out to a companion data warehouse
  - + Or HDFS or ....
  - + Here now
- If 90/10 rule is true
  - + Use “semantic caching”
  - + Subset of the rows are in memory. Transaction is stalled while needed data is loaded, then it is run
  - + Coming soon

# Current VoltDB Status

- Shipping V 2.0
- A dozen or so production customers
  - + That we know about.....
- A couple thousand downloads a month

# Next Steps

- WAN replication
- Semantic caching
- Compression (messages, data, command log)
- More SQL
- On-the-fly reprovisioning

# OldSQL vs NewSQL

- Disk vs main memory
- Dynamic locking vs something else
- Active-passive vs active-active
- Data log vs stored procedure log
- Slow vs fast