

# Urban Myths about SQL

Michael Stonebraker, CTO VoltDB

June 11, 2010





# Outline

- **6 Urban Myths**
  - Myth #1: SQL is too slow, so use a lower level interface
  - Myth #2: I like a K-V interface, so SQL is a non-starter
  - Myth #3: SQL systems don't scale
  - Myth #4: There are no open source, scalable SQL engines
  - Myth #5: ACID is too slow, so avoid using it
  - Myth #6: in CAP, choose AP over CA
- **Often based on VoltDB as a counterexample**
  - A pure SQL, pure ACID, shared nothing, sharded DBMS



## Myth #1: SQL is too slow

- **Fact**
  - All serious SQL engines compile either to intermediate representation or to C++ (or another programming language)
  - Hence, SQL is not slower than “rolling your own” in a lower level language
  - Remember the debate about assembly language vs a higher level language?????



# So Why Are Real SQL Systems Slow if It's not SQL?

- ODBC/JDBC
  - Really costly (open cursor, fetch, .... – each a round-trip message from client to server)
    - + Nobody serious uses it
  - Use stored procedures instead -- one message over and back
  - VoltDB supports only a stored procedure interface



# So Why Are Real SQL Systems Slow?

- Services (useful work is a small fraction of the total)
  - Disk buffer pool
  - Locking
  - Crash recovery
  - Multi-threading
- To go a lot faster, you need to get rid of this overhead
  - Traditional RDMS don't do this – and are pokey
  - But VoltDB does – and goes wildly faster



## On TPC-C ( An ACID benchmark)

- Single node performance
  - MySQL:  $X$
  - A very popular RDBMS elephant:  $2.5 * X$
  - VoltDB:  $100 * X$

Actually TPC-C like.....

I have gotten my hand slapped by TPC before



# Net-Net

- Performance has nothing to do with SQL
  - It is about the interface
  - And the implementation of services
- Traditional RDBMSs are slow
  - But that is not a SQL issue



## Myth #2: I Like a Get/Put K-V Interface, Why would I consider SQL?

- VoltDB supports a K-V interface as a thin layer on top of stored procedure SQL interface
  - Mystore (key, payload) is a table
  - Get/Put are stored procedures with parameters that do the obvious things
- Always an option to support a less functional interface on top of a more functional one
  - Can freely mix K-V access and SQL access



# The Proof is in the Pudding: a Benchmark

- The “game state” of a major game vendor
  - Currently K-V with a payload of 12K bytes
  - Supported by Memcached/MySQL
  - 50/50 gets and puts



## Benchmark Performance (single node)

- MySQL/Memcached:  $X$
  - Cassandra:  $7 * X$
  - VoltDB:  $15 * X$
- 
- And if you move to a multi-attribute schema, VoltDB gets increasingly faster than Cassandra ( $5+ X$ )



## Myth #3: SQL Systems Don't Scale

- **Fact**
  - Some SQL engines don't support multi-node operation (MySQL, Postgres, Oracle, SQLServer)
  - But lots do (DB2, Vertica, Asterdata, Greenplum, EnterpriseDB)
  - And the latter ones scale linearly on most workloads (see for example, paper by Abadi et. al. in SIGMOD 2009)



# VoltDB Scales Linearly on TPC-C

- Slope of the line is 0.9
- We have tested up to 100 cores on 12 nodes
- 300 cores coming soon



## Myth #4: There are no Open Source Scalable SQL DBMSs

- **Fact**
  - VoltDB is open source
  - Source code at [voltdb.com](http://voltdb.com)
  - Also the benchmark code for everything I have talked about
  - Have at it!!!!



## Myth #5: ACID is too Expensive, so Don't Support it.

- Some applications require ACID
  - E.g bank funds transfer
  - Not supporting real transactions just pushes the problem to user code, where it is MUCH harder to solve
- And others may need ACID in the future
  - Remember that DBMS applications live a very long time
  - And requirements may change
  - Future ACID means a fork lift upgrade.....
- For these people non-ACID is a non-starter



## Myth #5: ACID is too Expensive, so Don't Support it.

- Now to the “too expensive” part.....



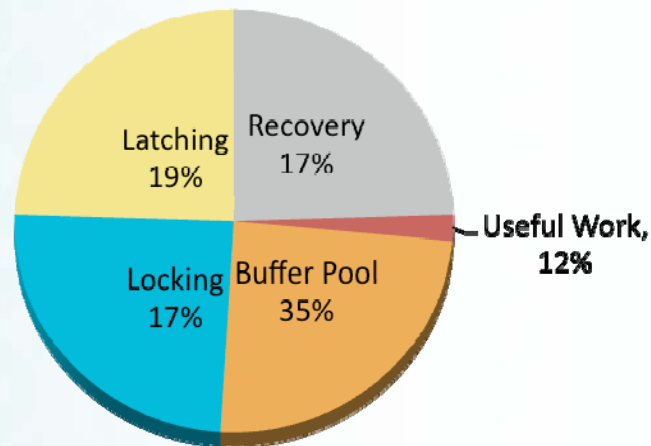
## A Key Assumption.....

- Most OLTP Databases are not huge
  - Factor out the trim (e.g. pictures) to somewhere else leaving only the transaction stuff
  - 1 Tbyte is a VERY big transaction data base
  - That is (or soon will be) a candidate for main memory deployment



# Reality Check (Harizopoulos et. al. SIGMOD 2008)

- TPC-C CPU cycles
- On the Shore DBMS prototype
- Most DBMSs should be similar





# Conclusions

- ACID is 1/3 of the cost
  - Deleting ACID wins 50%
- Possible to go 10X faster, but deleting ACID by itself won't do the trick
  - Buffer pool and multi-threading overhead are still onerous
- Better B-trees will do next to nothing!
- Have to get rid of all 4 sources of overhead to get blazing performance
  - VoltDB solution retains ACID and gets rid of all four sources of overhead



# CAP Theorem

- Can't have all three of:
  - Consistency (ACID)
  - Availability (HA)
  - Partition tolerance

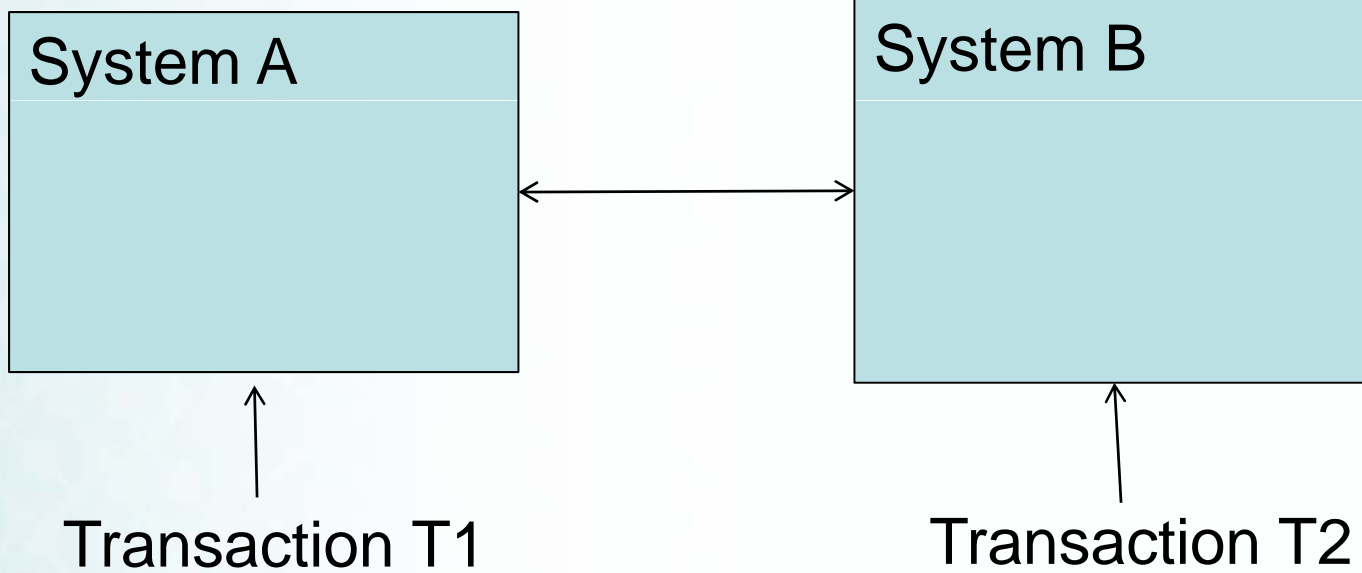


## Myth #6: Deleting C is a Good Idea

- This is something of a religious discussion....
- Three things to think about
  - Commutativity
  - Semantics
  - Errors

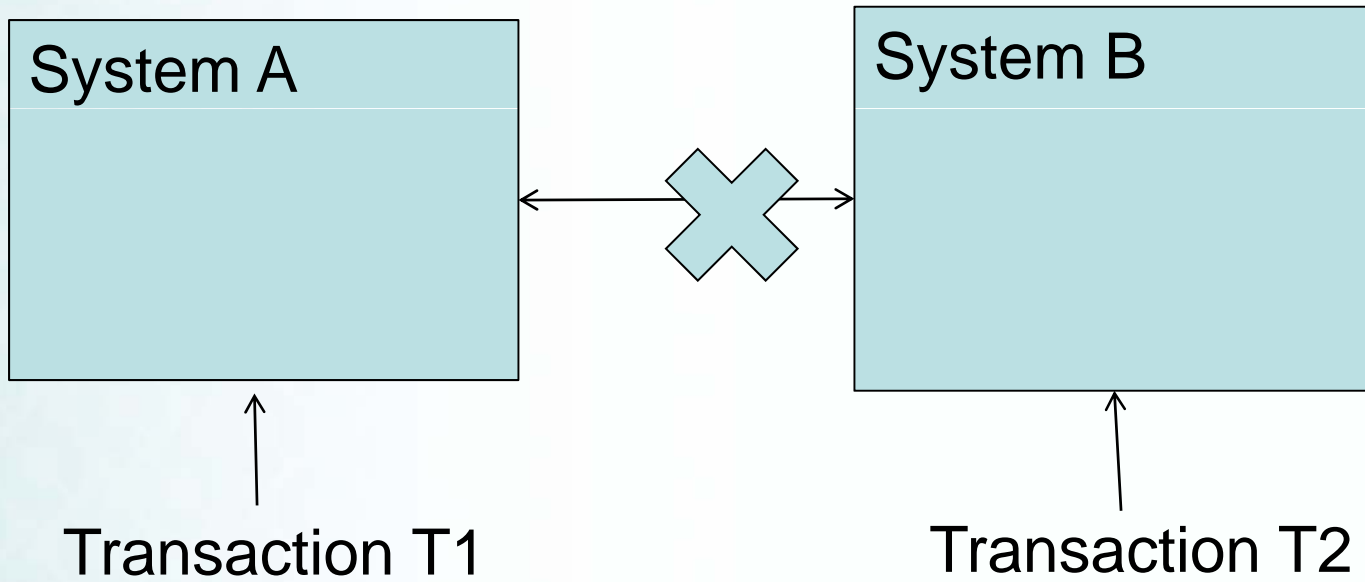


# Partition Behavior





# Partition Behavior





## Myth #6: Deleting C is a Good Idea

- Easy to get an order of execution of T1 then T2 at A and T2 and then T1 at B
- Consider the following
  - T1: give Mike a 10% raise
  - T2: increase Mike's salary \$500
- Starting with Mike's initial salary of \$10,000
  - T1 then T2: \$11,500
  - T2 then T1: \$11,550



## Myth #6: Deleting C is a Good Idea

- If your transactions are not commutative, then you get garbage from partition tolerance and eventual consistency
- Have to guarantee commutativity now
  - And forever more – remember the forklift!



## Myth #6: Deleting C is a Good Idea

- Consider ordering parts from a warehouse
  - Suppose there is a constraint that you can't accept an order if stock is depleted
  - Impossible to support such semantic constraints with partition tolerance and eventual consistency



## Myth #6: Deleting C is a Good Idea

- Errors that can cause outages
  - Human error – a human screwed up (both copies of) the data base
  - Application error – application screwed up (both copies of) the data base
  - Bohr bug in the DBMS – DBMS screwed up (both copies of the data base)
  - Data base taken off line for an upgrade, fork-lift upgrade, or resource issue (somebody ran out of memory)
  - Non-trivial network partition (something other than N-1 and 1)



# Conclusion

- Network partitions do not appear to be the “high pole in the tent”
  - Giving up C all the time
  - In exchange for letting both sides of a partition continue (as opposed to just one) when a rare event happens
  - Seems like a poor tradeoff!
- Clearly depends on outage distribution
  - I am currently doing a study with a major, world wide, financial institution.
  - Pouring over and classifying their error logs



# And Now A Word From Our Sponsor....

## VoltDB in a Nutshell

- Main-memory storage
  - No buffer pool (one of the four horsemen)
- Run transactions to completion – single threaded – in timestamp order
  - No disk stalls; no human stalls inside a Xact
  - OLTP Xacts are lightweight
  - No locking, no latching (2 and 3 are gone)
  - But ACID retained



# VoltDB in a Nutshell

- Built-in HA/DR
  - Based on replicas
  - No log (4<sup>th</sup> horseman bites the dust)
- Replicas are done active-active
  - In the same timestamp order
  - Consistency; not eventual consistency



# VoltDB Status

- V1 is available
  - TCP/IP, Linux, hardware-agnostic
  - About 50 users at the current time
  - 4 close to production
- Pure SQL
  - Subset of SQL '99
  - On LAN clusters
  - No WAN support yet
- Pure ACID; CA; not AP



# Thank You

- For more information on VoltDB:

[www.VoltDB.com](http://www.VoltDB.com)