Feb 2015

NewSQL Overview

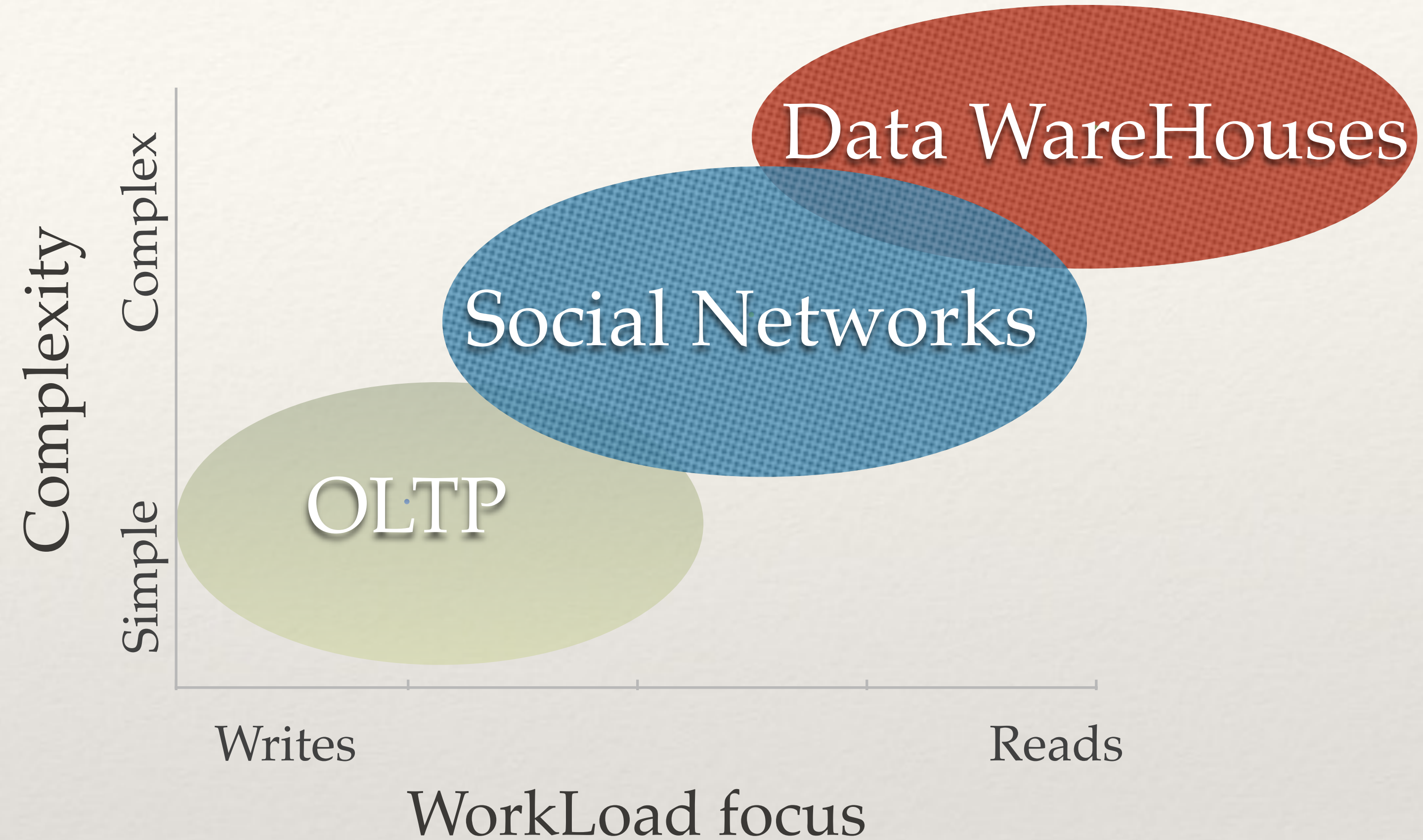
Ivan Glushkov
@gliush
ivan.glushkov@gmail.com

About myself

- MIPT
- MCST, Elbrus compiler project
- Echo, real-time social platform (PaaS)
- DevZen podcast (http://devzen.ru)

History of SQL

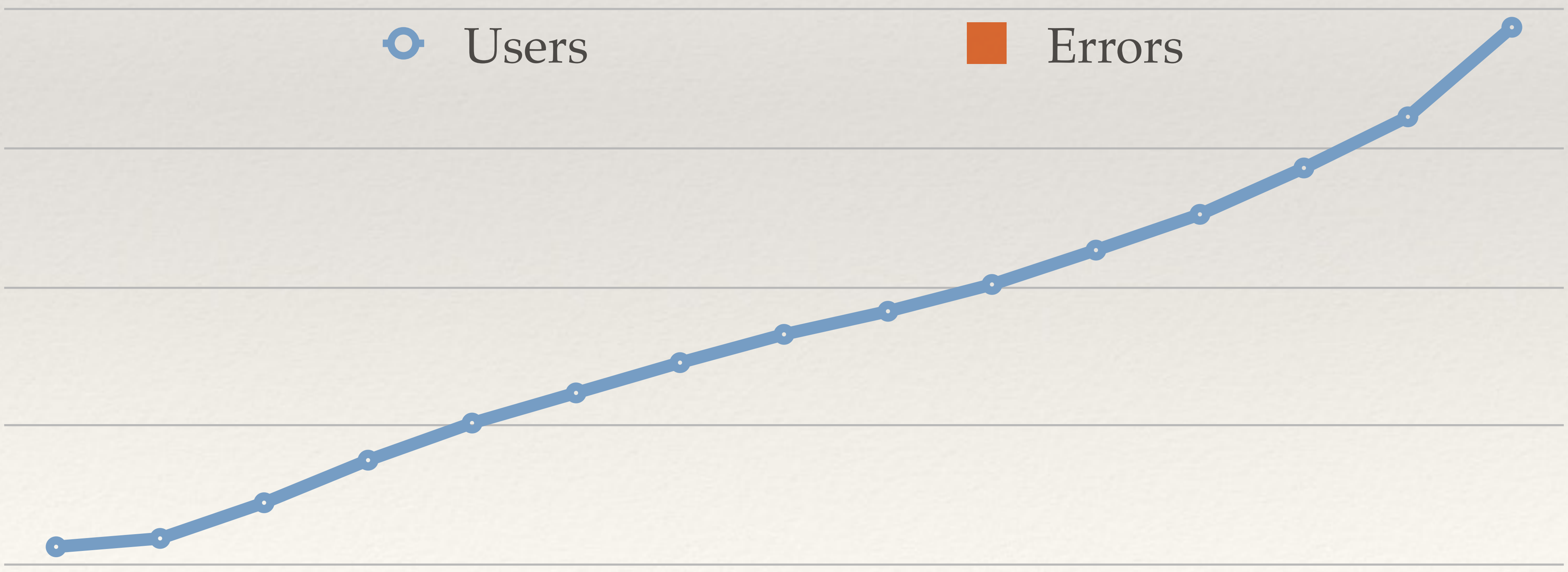
- Relational Model in 1970
  - disk-oriented
  - rows
  - sql



- "One size fits all" doesn't work:
  - Column-oriented data warehouses for OLAP.
  - Key-Value storages, Document storages

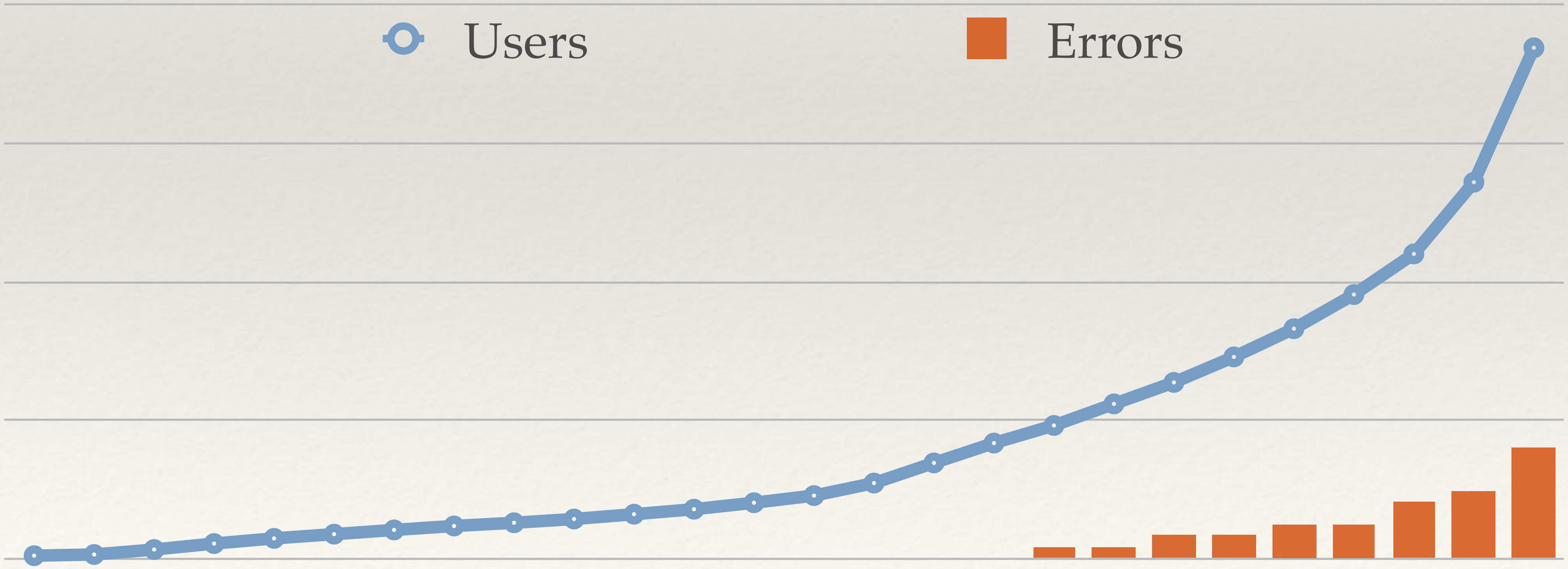
Startups lifecycle

- Start: no money, no users, open source



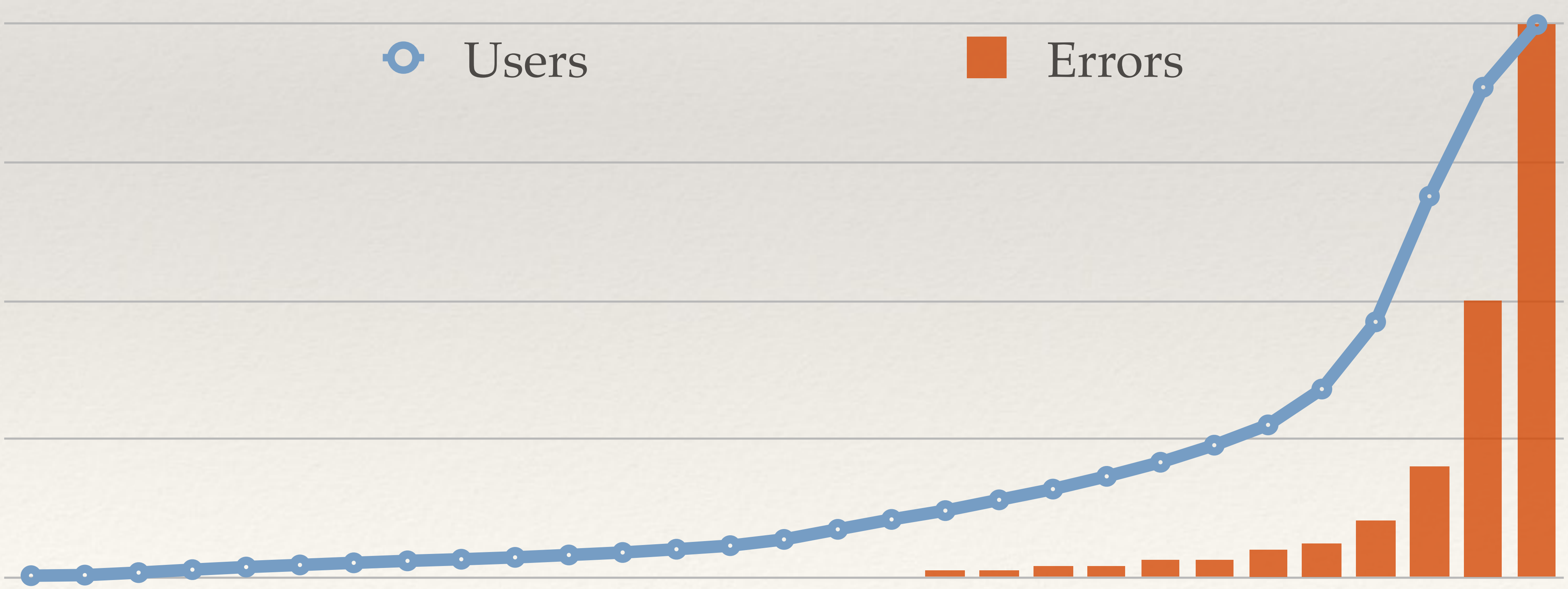
Startups lifecycle

- Start: no money, no users, open source
- Middle: more users, storage optimization



Startups lifecycle

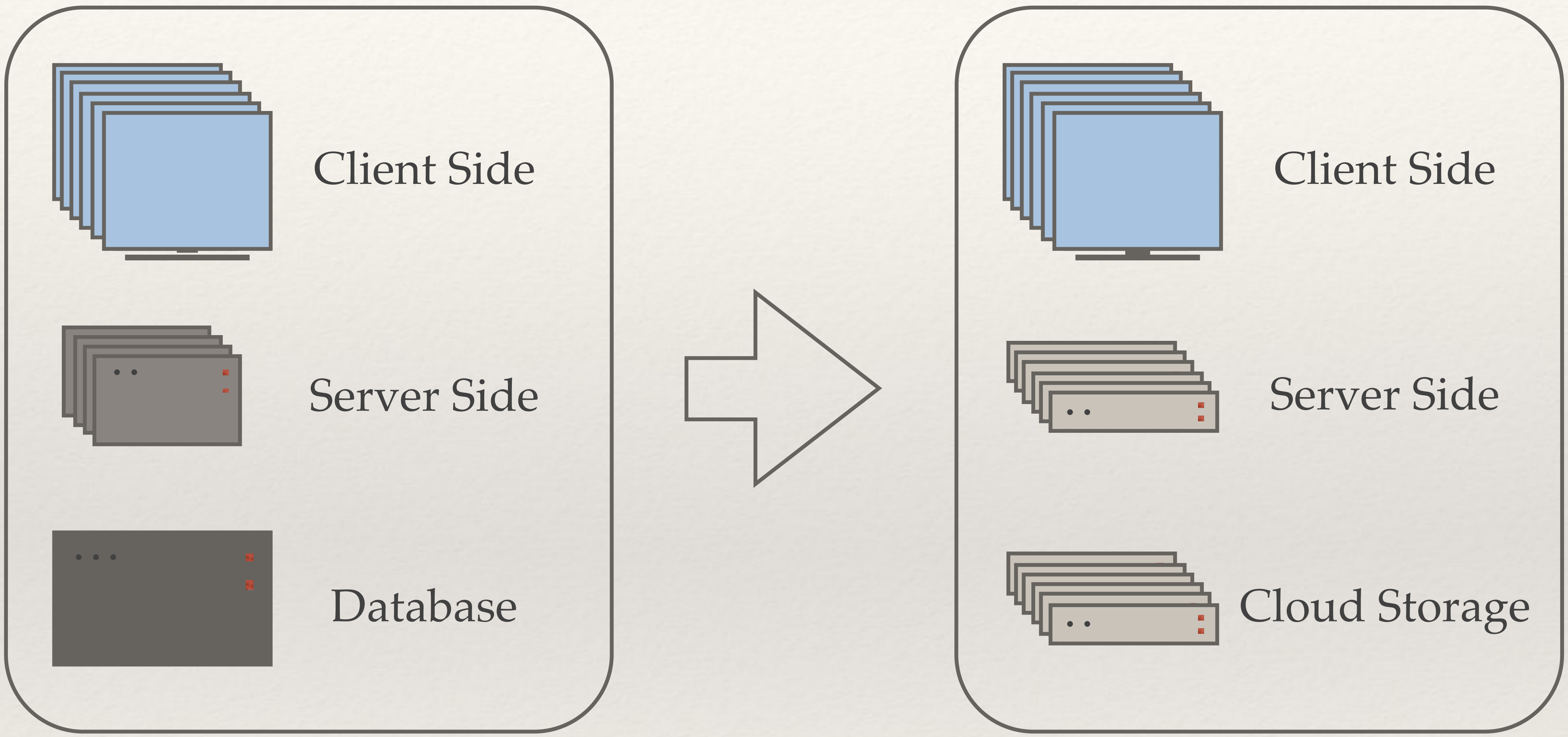
- Start: no money, no users, open source
- Middle: more users, storage optimization
- Final: plenty of users, storage failure



New requirements

- Large scale systems, with huge and growing data sets
  - 9M messages per hour in Facebook
  - 50M messages per day in Twitter
- Information is frequently generated by devices
- High concurrency requirements
- Usually, data model with some relations
- Often, transactional integrity

Trends: architecture change



Consistency, transactions: Database

Storage optimization: Database

Scalability: Client Side

Consistency, transactions: Cloud

Storage optimization: Cloud

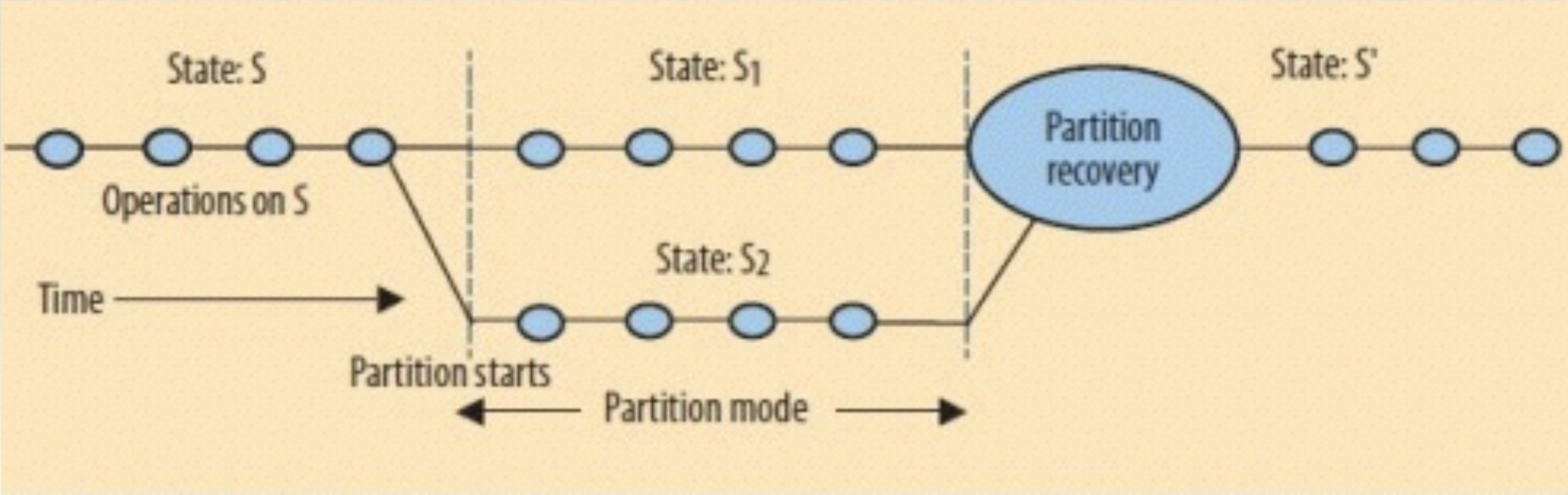
Scalability: All levels

Trends: architecture change

- CAP: **c**onsistency, **a**vailability, **p**artitioning
- ACID: **a**tomicity, **c**onsistency, **i**solation, **d**urability
- BASE: **b**asically **a**vailable, **s**oft state, **e**ventual consistency

Trends: architecture change

- 'P' in CAP is not discrete
- Managing partitions: detection, limitations in operations, recovery



NoSQL

- CAP: first ‘A’, then ‘C’: finer control over availability
- Horizontal scaling
- Not a “relational model”, custom API
- Schemaless
- Types: Key-Value, Document, Graph, …

Application-level sharding

- Additional application-level logic
- Difficulties with cross-sharding transactions
- More servers to maintain
- More components — higher prob for breakdown

NewSQL: definition

"A DBMS that delivers the scalability and flexibility promised by NoSQL while retaining the support for SQL queries and/or ACID, or to improve performance for appropriate workloads."

451 Group

NewSQL: definition

- *SQL as the primary interface*
- *ACID support for transactions*
- *Non-locking concurrency control*
- *High per-node performance*
- *Scalable, shared nothing architecture*

Michael Stonebraker

Shared nothing architecture

- No single point of failure
- Each node is independent and self-sufficient
- No shared memory or disk
- Scale infinitely
- Data partitioning
- Slow multi-shards requests

Column-oriented DBMS

- Store content by column rather than by row

John	Smith	20
Joe	Smith	30
Alice	Adams	50

⇨

John:001; Joe:002; Alice:003.

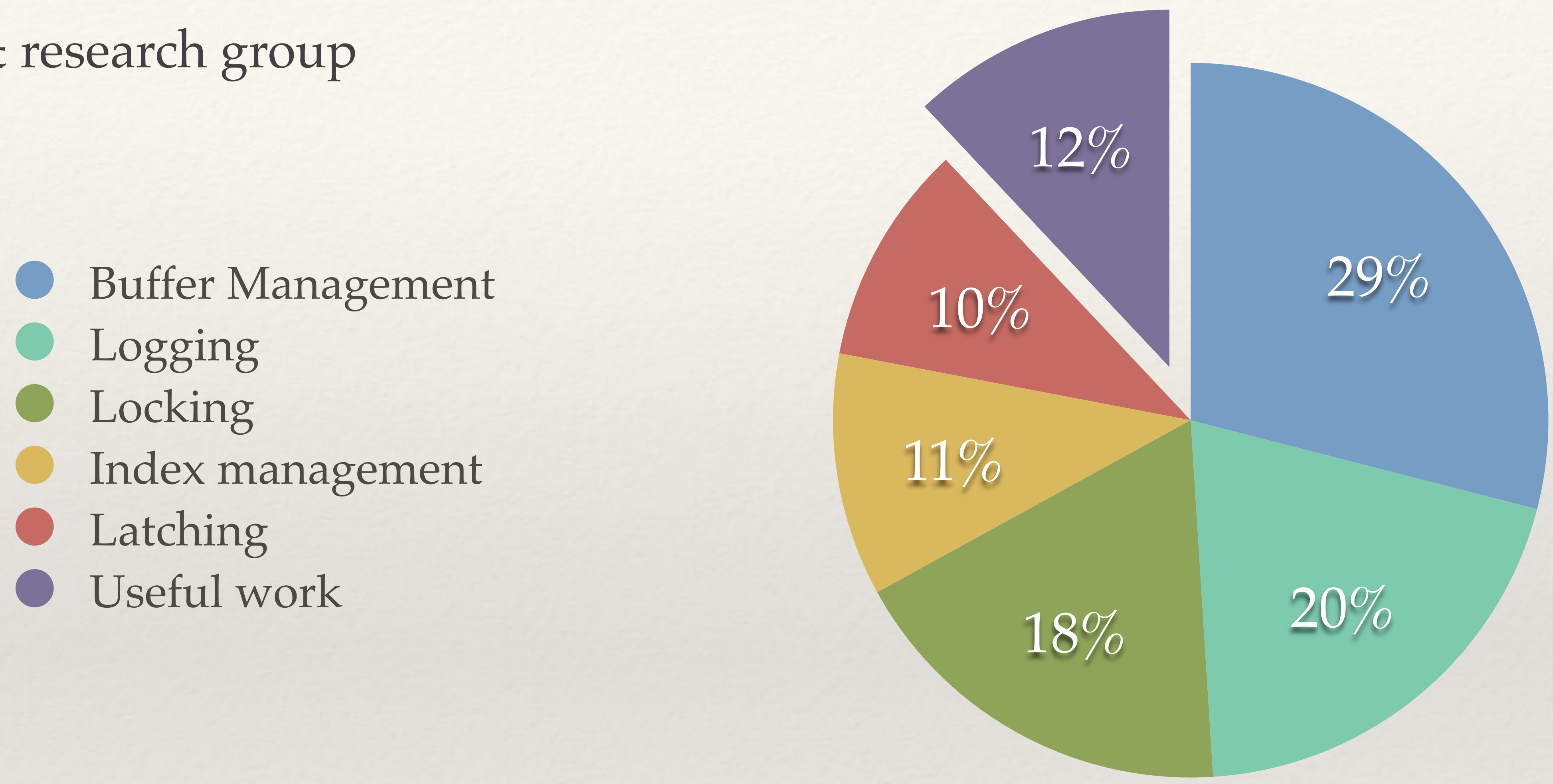
Smith:001,002; Adams:003.

20:001; 30:002; 50:003.

- Efficient in hard disk access
- Good for sparse and repeated data
- Higher data compression
- More reads/writes for large records with a lot of fields
- Better for relatively infrequent writes, lots of data throughput on reads (OLAP, analytic requests).

Traditional DBMS overheads

by Stonebraker & research group



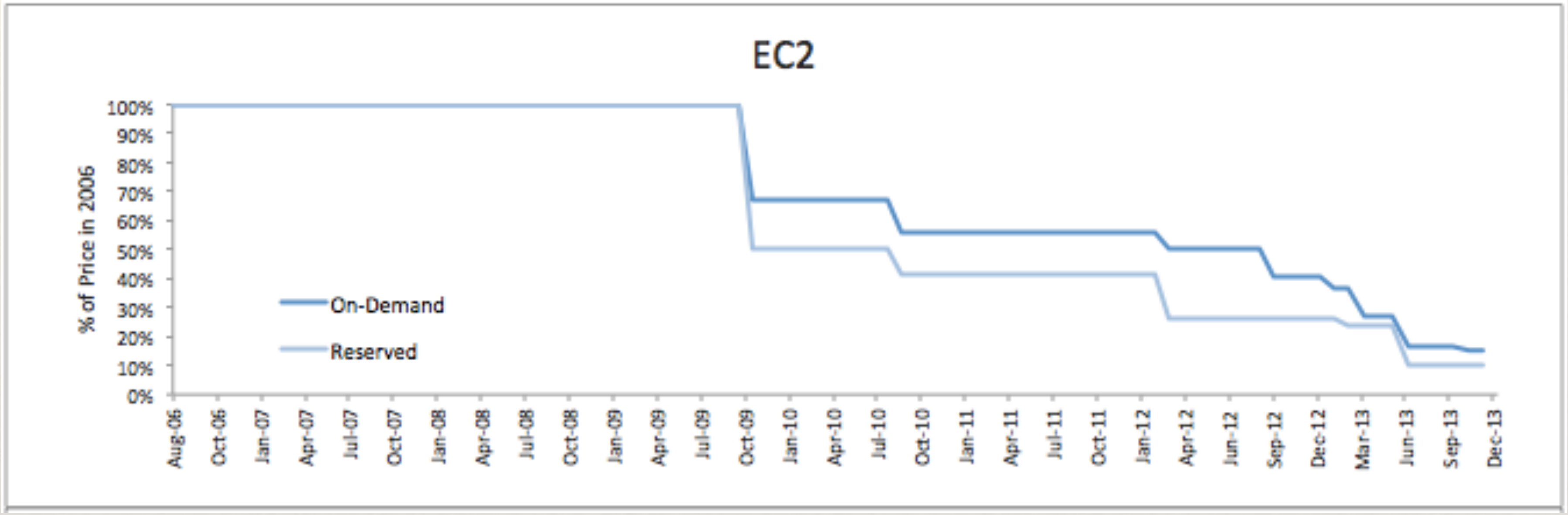
"Removing those overheads and running the database in main memory would yield orders of magnitude improvements in database performance"

In-memory storage

- High throughput
- Low latency
- No Buffer Management
- If serialized, no Locking or Latching

In-memory storage: price

Amazon price reduction



Current price for 1TB (~4 instances of ‘r3.8xlarge’ type)

	on-demand	3Y-reserved plan
per hour	11.2 $	3.9 $
per month	8.1K $	2.8K $
per year	97K $	33,7K $

NewSQL: categories

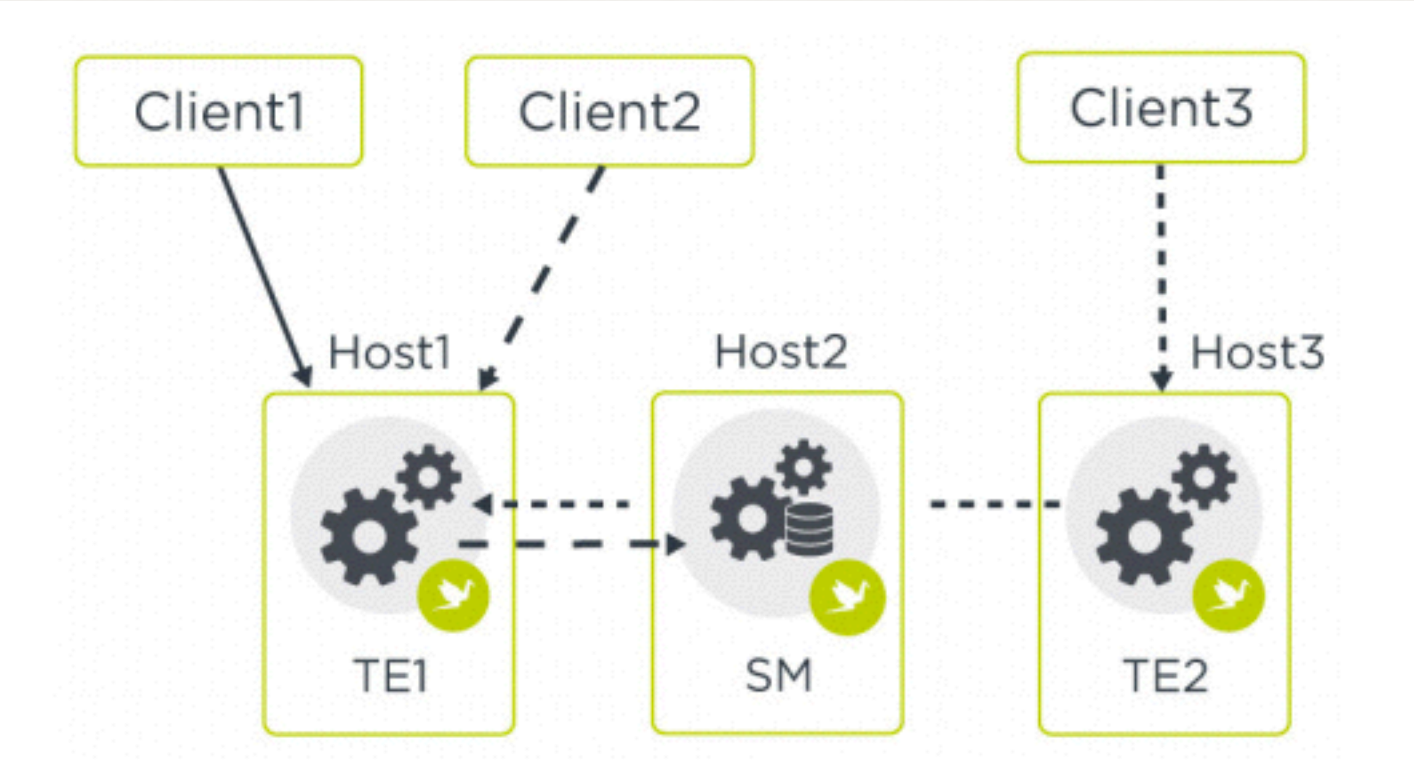
- New approaches: VoltDB, Clustrix, NuoDB
- New storage engines: TokuDB, ScaleDB
- Transparent clustering: ScaleBase, dbShards

NuoDB

- Multi-tier architecture:
    - Administrative: managing, stats, cli, web-ui
    - Transactional: ACID except ‘D’, cache
    - Storage: key-value store (‘D’ from ACID)

NuoDB

- Everything is an ‘Atom’
- Peer-to-peer communication, encrypted sessions
- MVCC + Append-only storage

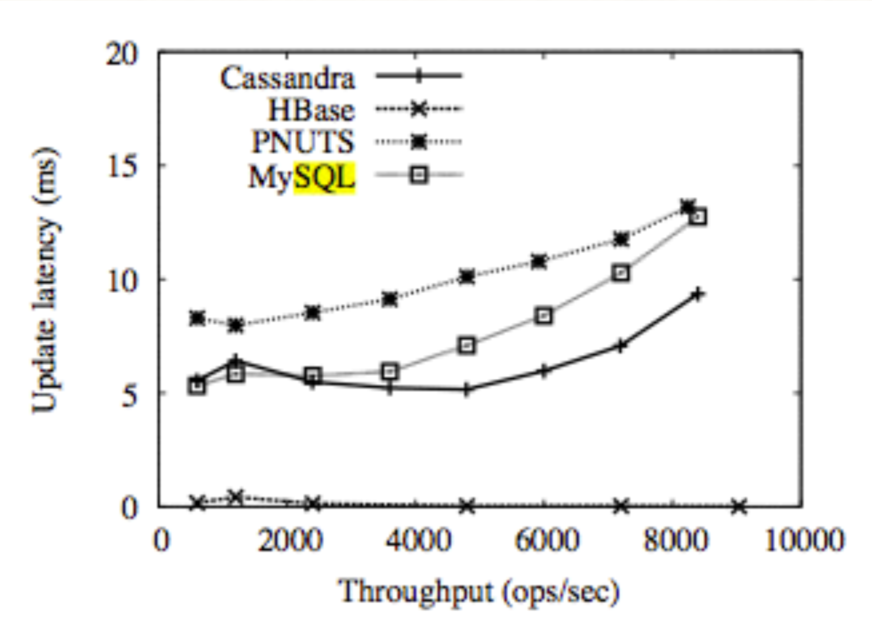


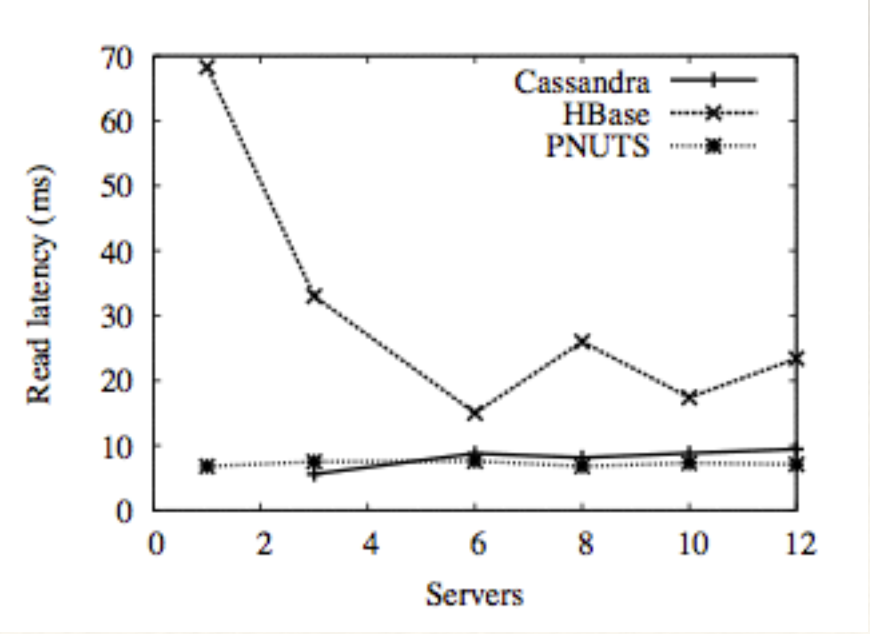
NuoDB: CAP & ACID

- `CP` system. Need majority of nodes to work
- If split to two equal parts -> stop
- Several consistency modes including ‘consistent_read’

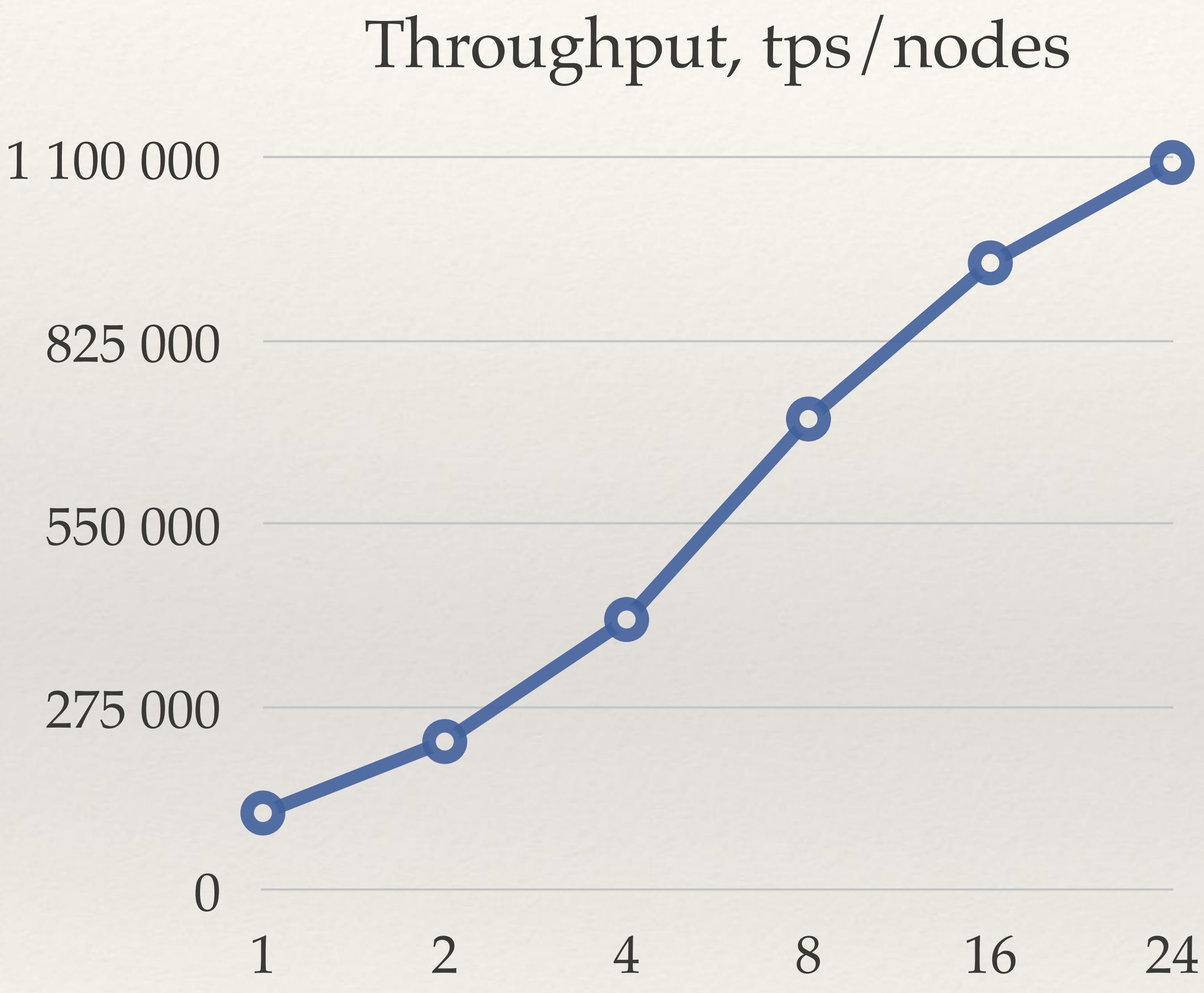
YCSB

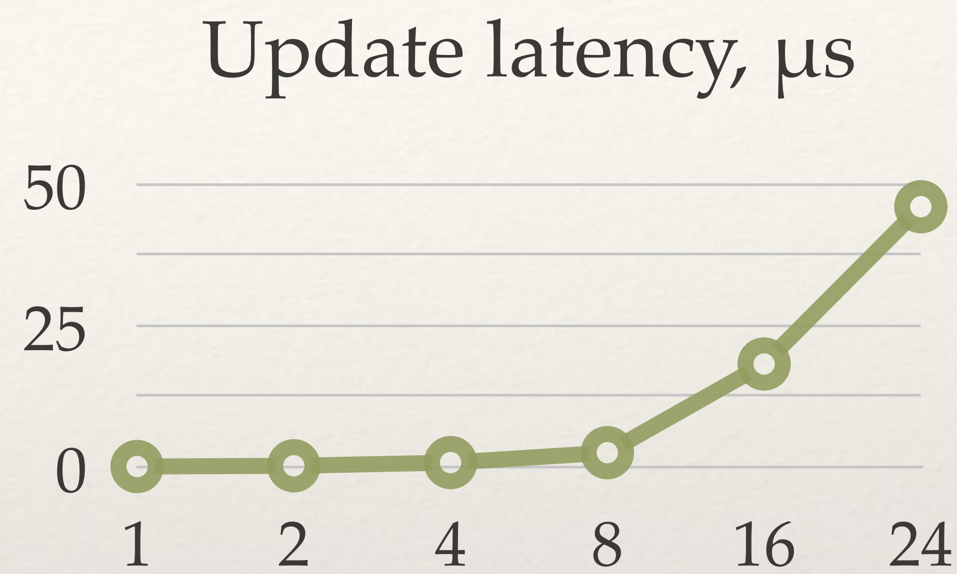
- Yahoo Cloud Serving Benchmark
- Key-value: insert / read / update / scan
- Measures:
  - Performance: latency / throughput
  - Scaling: elastic speedup

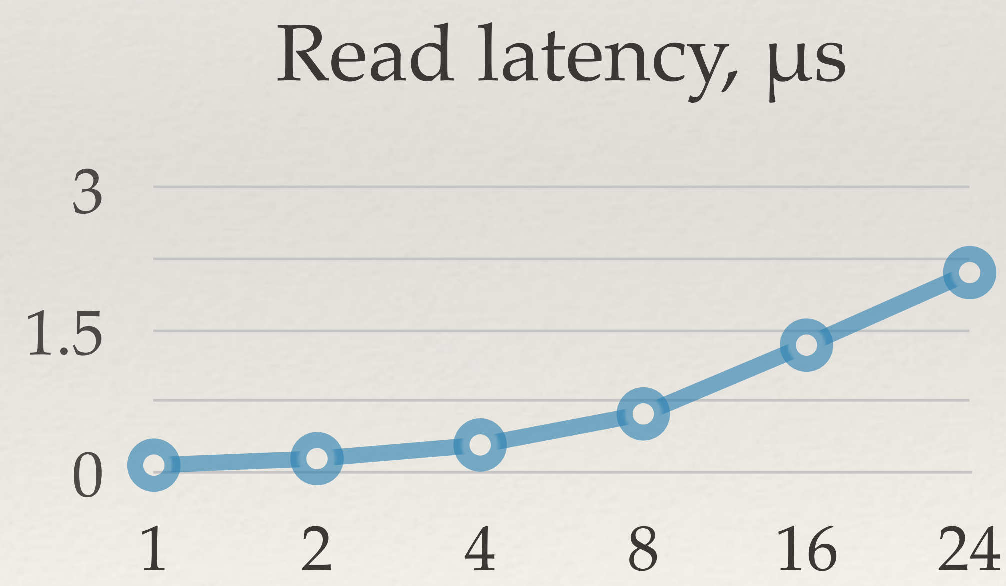




NuoDB: YCSB







5% updates, 95% reads

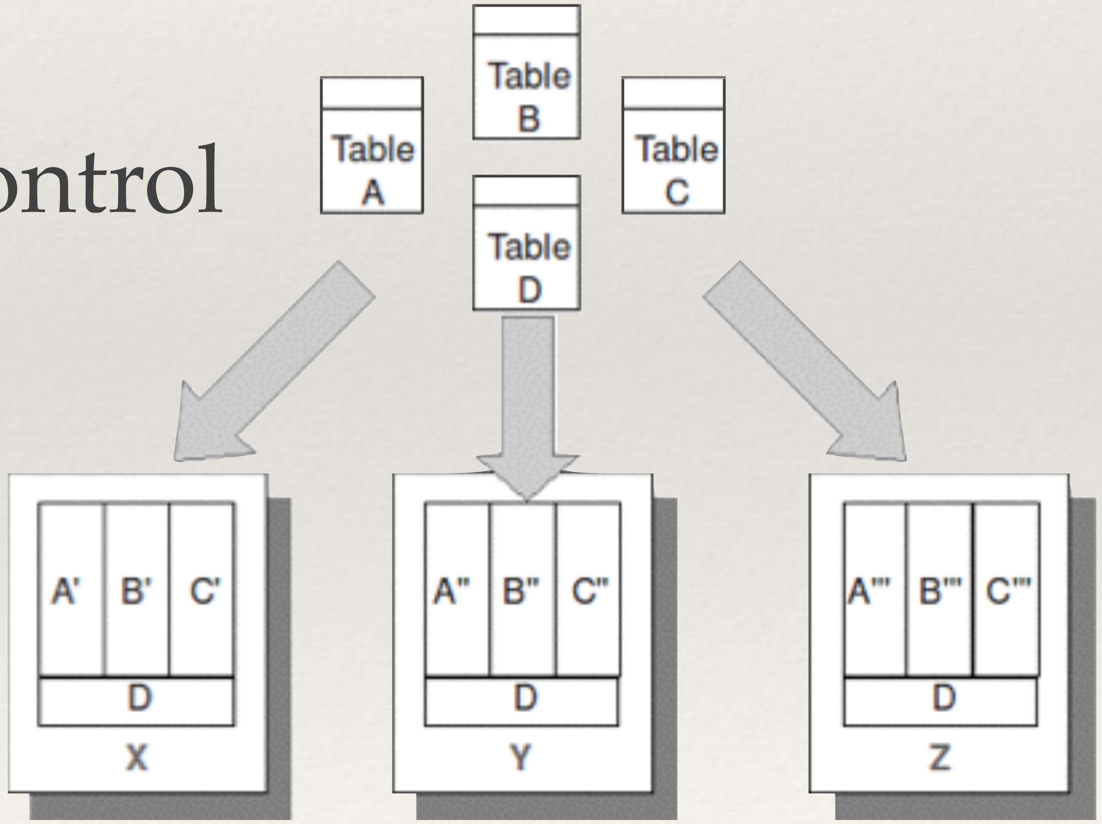
Hosts: 32GB, Xeon 8 cores, 1TB HDD, 1Gb LAN

VoltDB

- In-memory storage
- Stored procedure interface, async/sync proc execution
- Serializing all data access
- Horizontal partitioning
- Multi-master replication ("K-safety")
- Snapshots + Command Logging

VoltDB

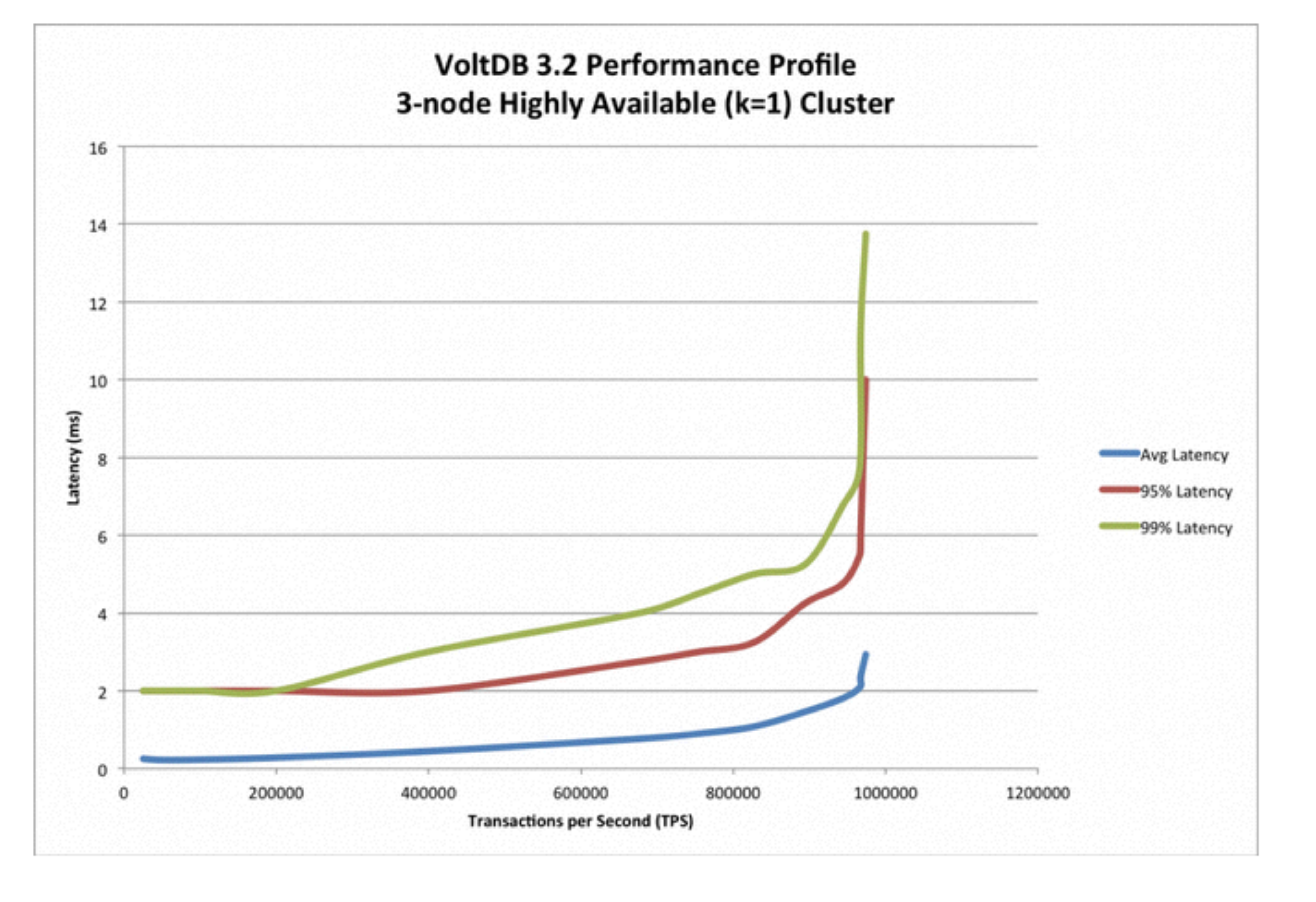
- Open-source, community edition is under GPLv3.
- Java + C++
- Partitioning and Replication control



VoltDB: CAP & ACID

- Without K-safety, any node fail break the whole DB
- Snapshot and shutdown minor segments during network paritions
- Single-partition transactions are very fast
- Multi-partition transactions are slower (manager), try to avoid (1000s tps in ’13, no updates since)

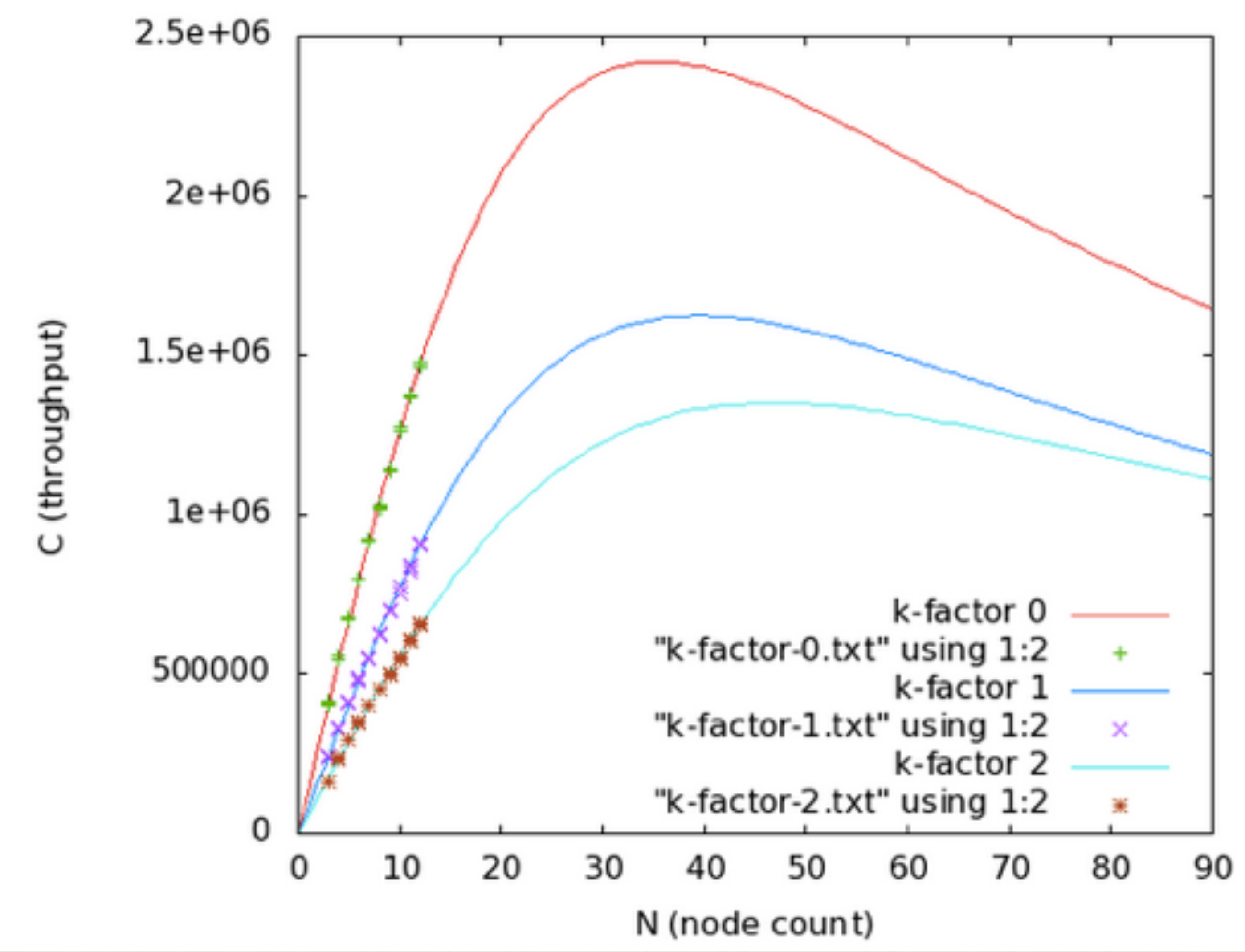
VoltDB: key-value bench



90%reads, 10%writes

3 nodes: 64GB, dual 2.93GHz intel 6 core processors

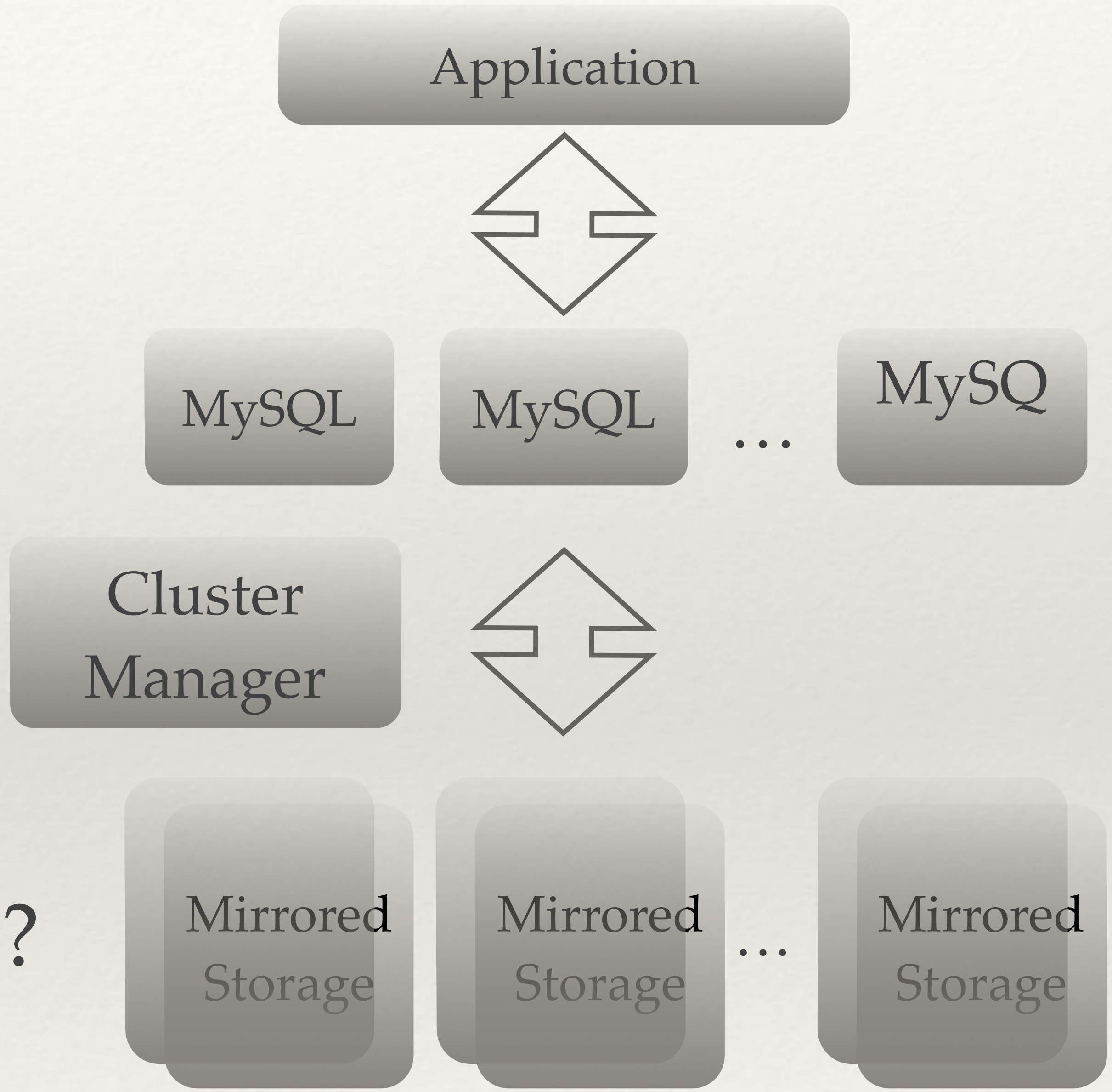
VoltDB: “voter” bench



26 SQL statements per transaction

ScaleDB

- Multi-master
- Shared data
- Cluster manager to solve conflicts (locks)
- ACID?
- Network Partition Handling?
- Scaling?



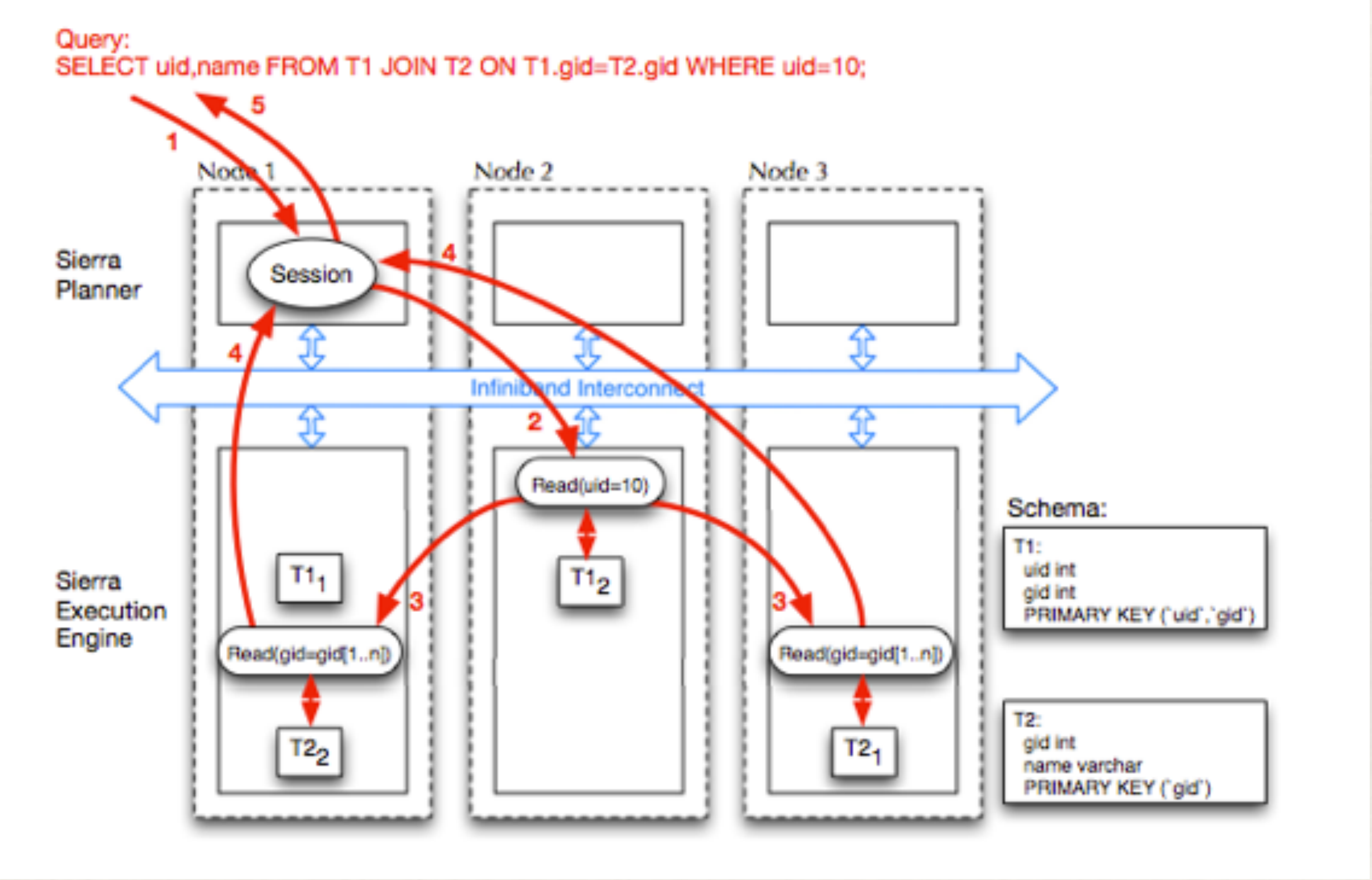
ClustrixDB

- "Query fragment" - basic primitive of the system:
  - read / write / execute function
  - modify control flow
  - perform synchronisation
  - send rows to query fragments on another nodes
- Data partitions: "slices" split and moved transparently
- Replication: master slice for reads + slave for redundancy

ClustrixDB

Clustrix

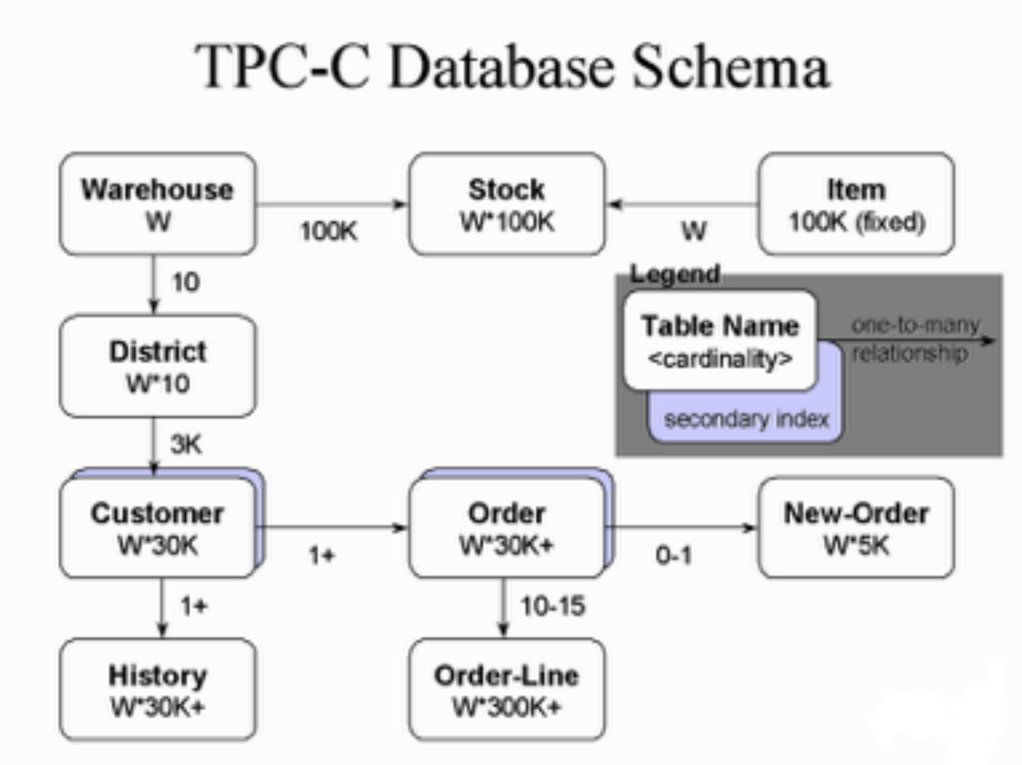
- “Move query to the data”
- Dynamic and transparent data layout
- Linear scale



ClustrixDB: CAP & ACID

- `CP` system. Need majority of nodes to work
- Only ‘Repeatable Read’ isolation level
  (so, ‘fantom reads’ are possible)
- Distributed Lock Manager for writer-writer locks (on each node)

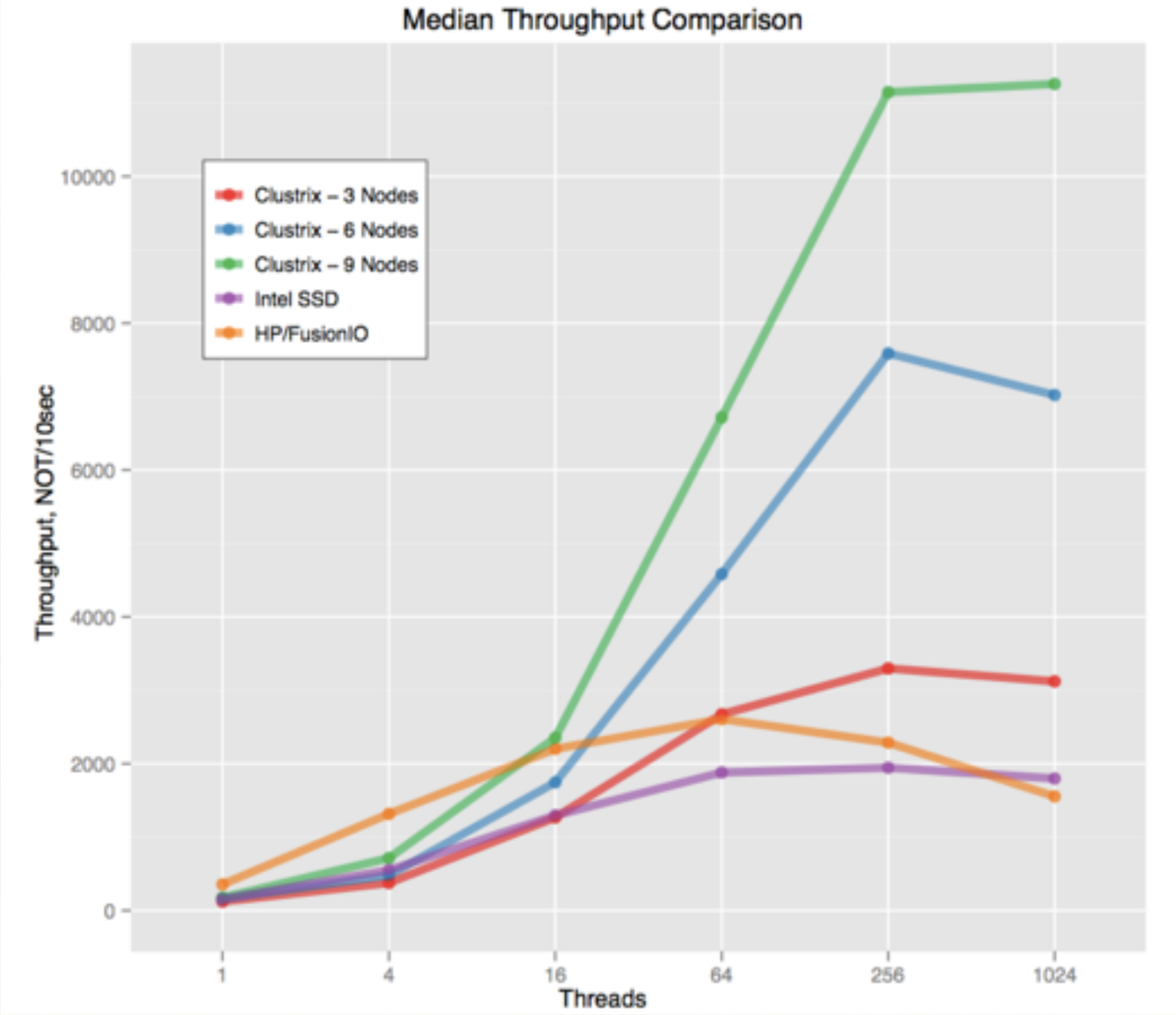
TPC-C



- Online Transaction Processing (OLTP) benchmark
- 9 types of tables
- 5 concurrent transactions of different complexity
- Productivity measured in "new-order transaction"

ClustrixDB: TPC-C

- 5000W ~ 400GB of data
- Compared with Percona Mysql, Intel Xeon, 8 cores
- ClustrixDB nodes: "Dual 4 core Westmere processors"



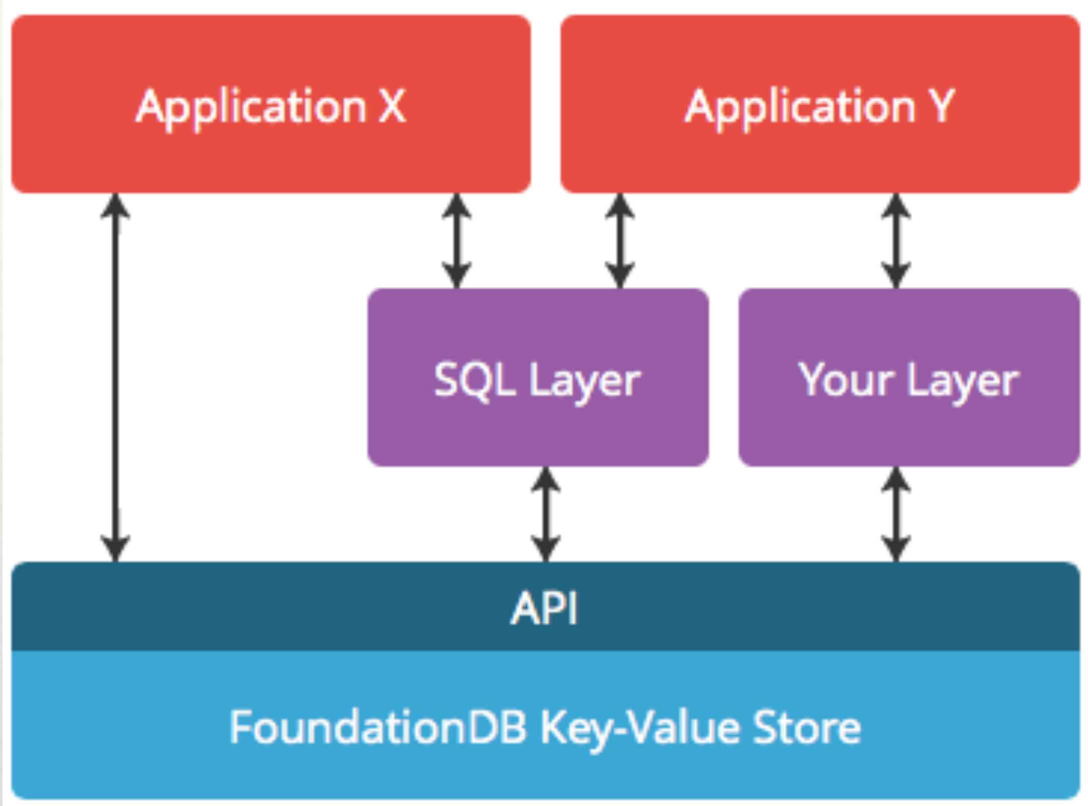
ClustrixDB: example

- 30M users, 10M logins per day
- 4.4B transactions per day
- 1.08/4.69 Petabytes per month writes/reads
- 42 nodes, 336 cores, 2TB memory, 46TB SSD

FoundationDB

- KV store, ordered keys
- Paxos for cluster coordination
- Global ACID transactions, range operations
- Lock-free, optimistic concurrency, MVCC
- Good testing (deterministic simulation)
- Fault-tolerance (replication)
- SQL Layer (similar to Google F1 on top of Spanner)

FoundationDB



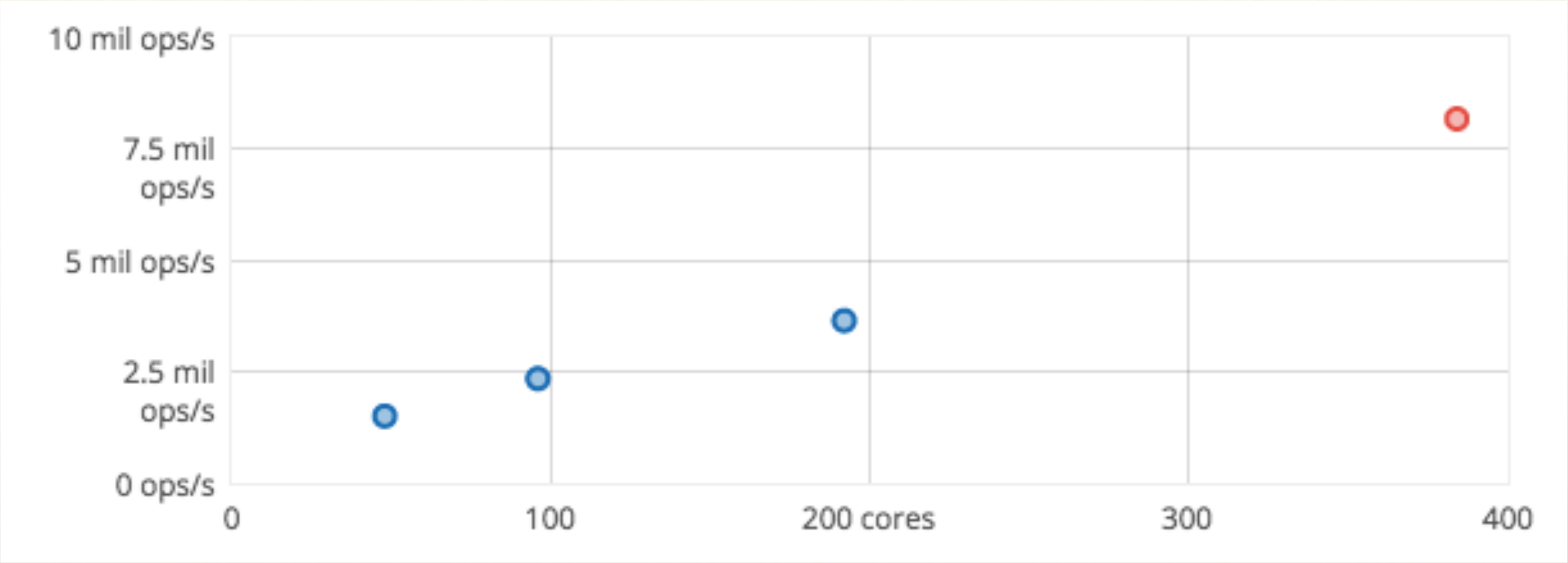
- SSD/Memory storage engine
- Layers concept
- 'CP' system with Paxos-ed coordination centres
- Written in the Flow language (translated to C++11) with actor model support
- Watches, atomic operations (e.g. 'add')

FoundationDB: CAP and ACID

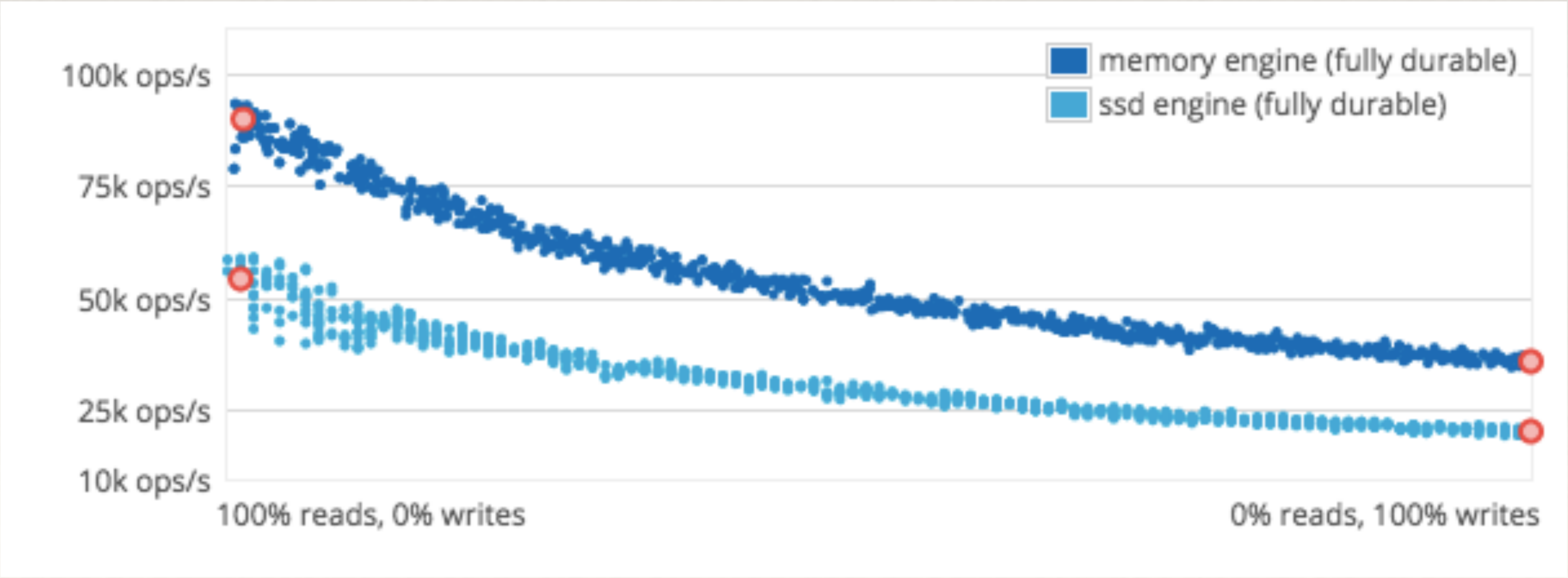
- Serializable isolation with optimistic concurrency
- > 100 wps to the same key? Use another DB!
- ‘CP system’ (Paxos)
  Need majority of coordination center to work

FoundationDB: KV Performance

Scaling:
up to 24 EC2 c3.8xlarge, 16 cores

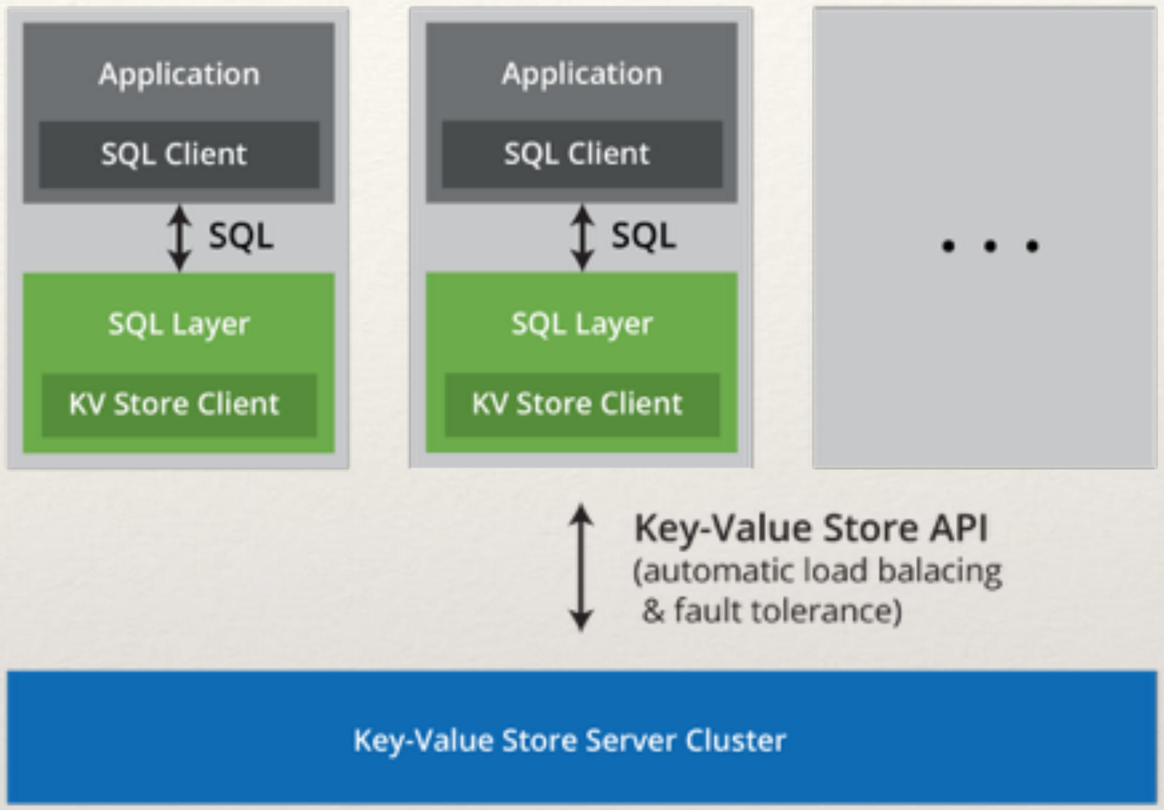


Throughput (per core)



FoundationDB:SQL Layer

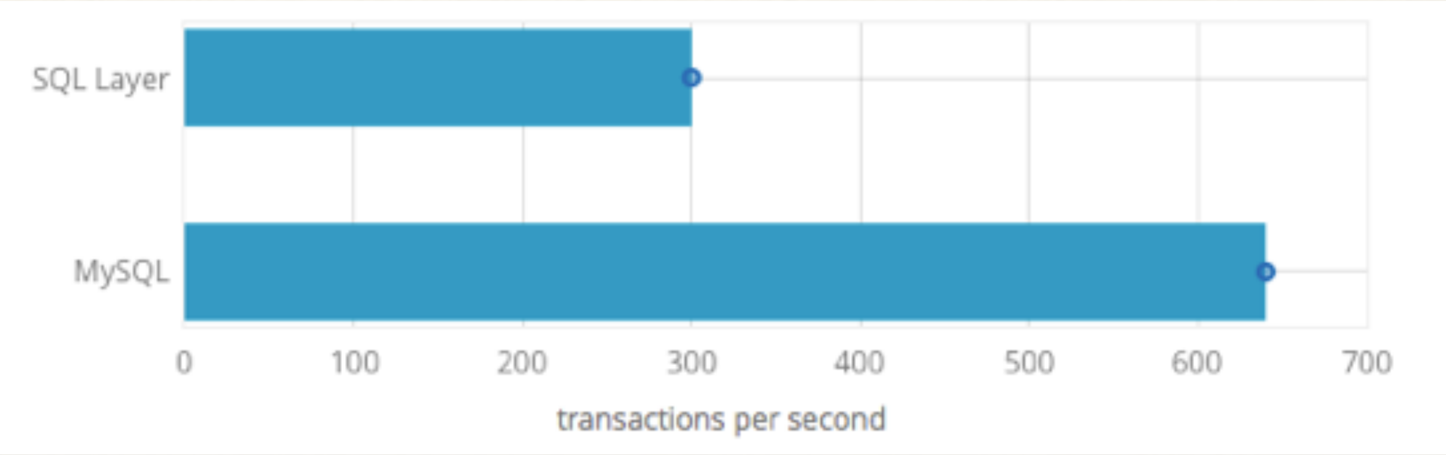
- SQL - layer on top of KV -> transactional, scalable, HA
- SQL Layer is stateless -> scalable, fault tolerant
- Hierarchical schema
- SQL and JSON interfaces
- Powerful indexing (multi-table, geospatial, …)



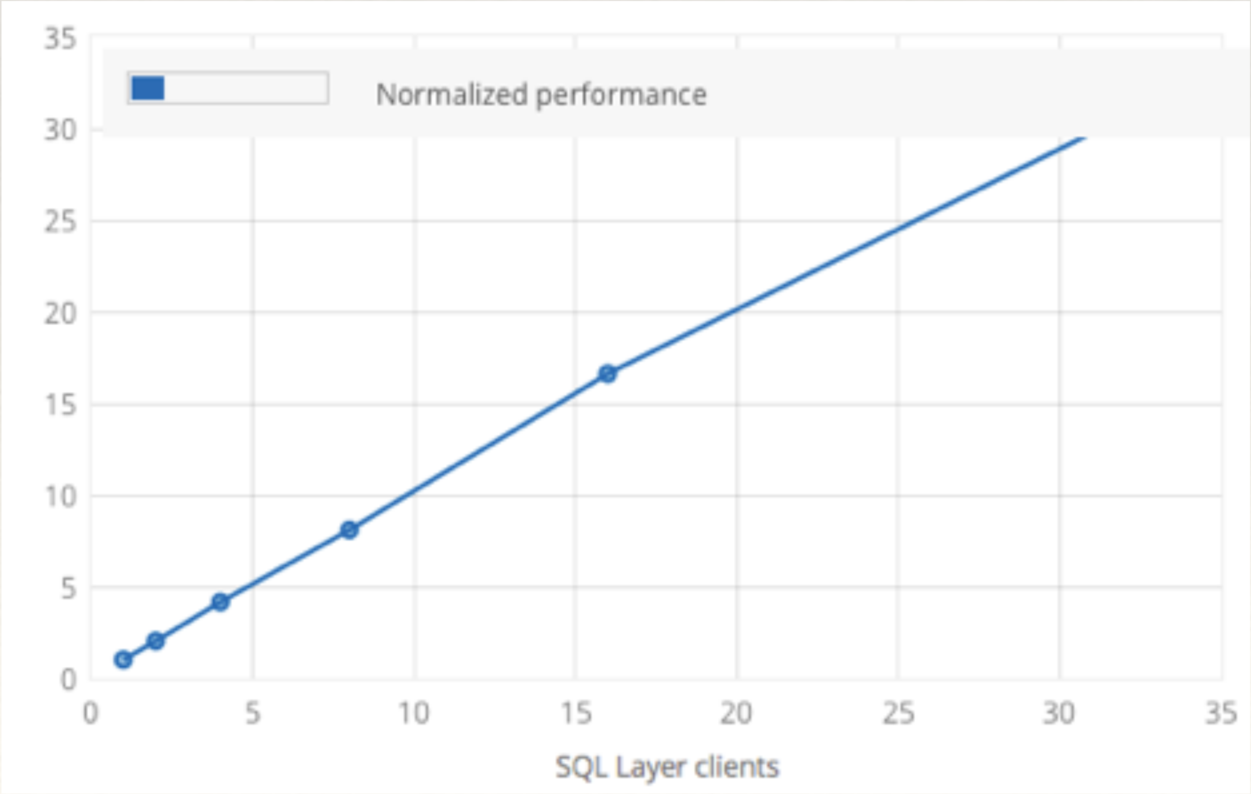
FoundationDB: SQL Performance

Sysbench: read/write, ~80GB, 300M rows

One node test
4 core, 16GB RAM, 200GB SATA SSD



Multi nodes test
KV: 8 nodes with 1-process; 3-replication
SQL: up to 32 nodes with
8-thread sysbench process

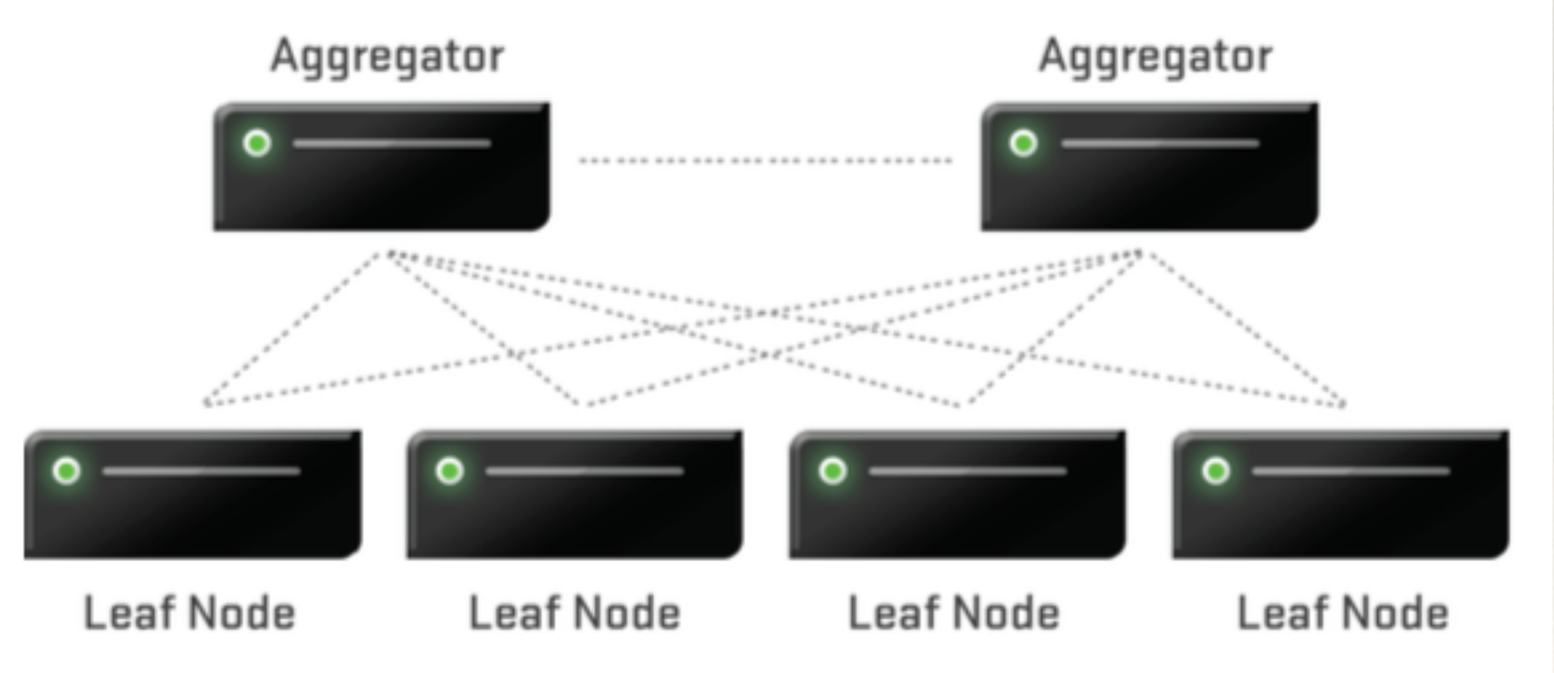


MemSQL

- In-Memory Storage for OLTP
- Column-oriented Storage for OLAP
- Compiled Query Execution Plans (+cache)
- Local ACID transactions (no global txs for distributed)
- Lock-free, MVCC
- Fault tolerance, automatic replication, redundancy (=2 by default)
- [Almost] no penalty for replica creation

MemSQL

- Two-tiered shared-nothing architecture
  - Aggregators for query routing
  - Leaves for storage and processing
- Integration:
  - SQL
  - MySQL protocol
  - JSON API

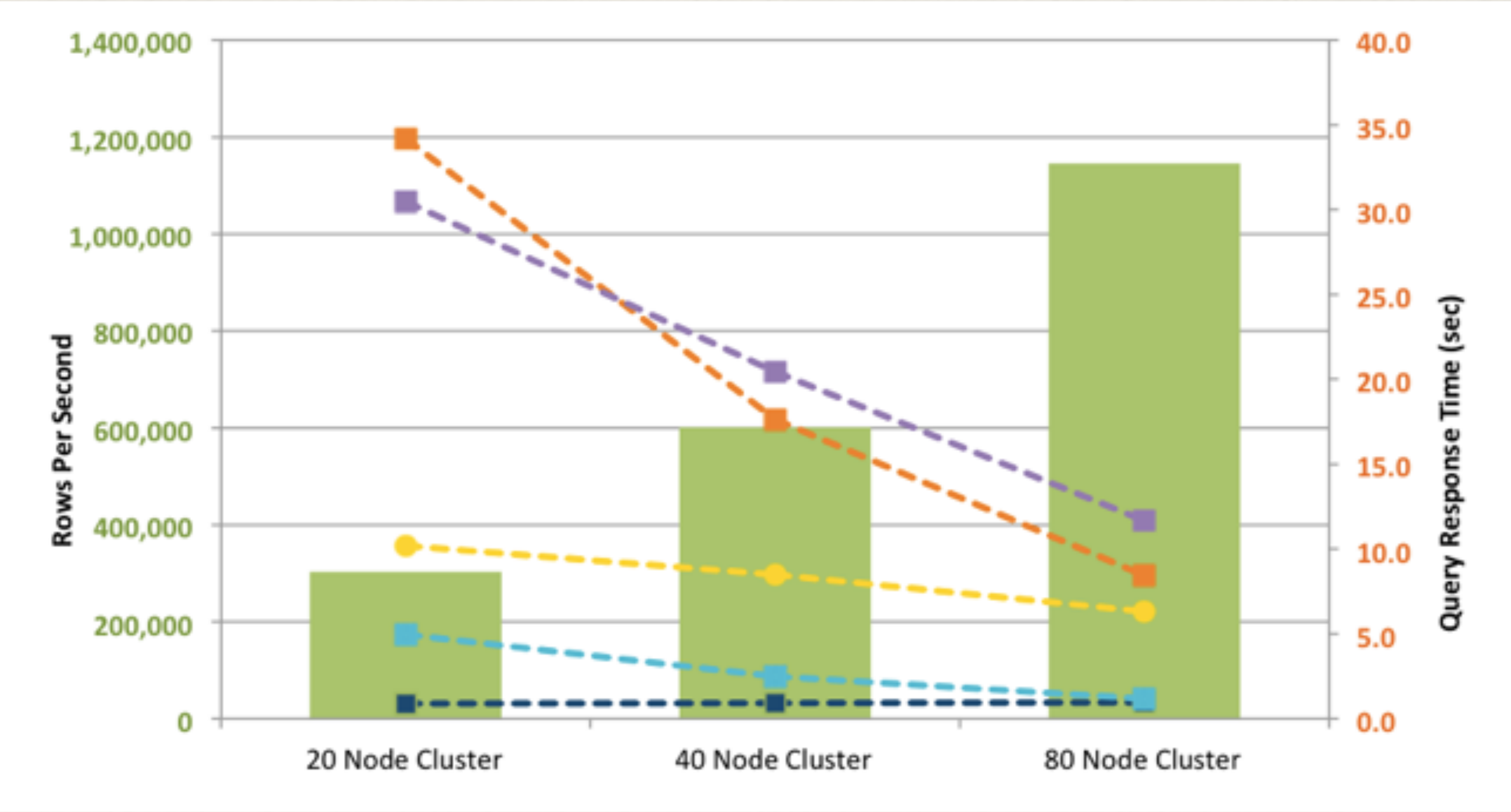


MemSQL: CAP & ACID

- `CP` system. Need majority of nodes (or half with master) to work
- Only 'Read Committed' isolation level ('fantom reads', 'non-repeatable reads' are possible)
- Manual Master Aggregator management

MemSQL: Performance

- Adapted TPC-H
- OLAP Reads & OLTP writes simultaneously
- AWS EC2 VPC



Overview

	Max Isolation	Scalable	Open Source	Free to try	Language
PostgreSQL	S	Postgres-XL?	Yes	Yes	C
NuoDB	CR	Yes	No	<5 domains	C++
VoltDB	S	Yes	Yes	Yes (wo HA)	Java/C++
ScaleDB	RC?	Yes?	No	?	?
ClustrixDB	RR	Yes	No	Trial (via email req)	C ?
FoundationDB	S	Yes	Partly	<6 processes	Flow(C++)
MemSQL	RC	Yes	No	?	C++

S: Serializable, RR: Read Committed, RC: Read Committed, CR: Consistent Read

Conclusions

- NewSQL is an established trend with a number of options
- Hard to pick one because they're not on a common scale
- No silver bullet
- Growing data volume requires ever more efficient ways to store and process it

Questions?

Links: General concepts

- CAP explanation from Brewer, 12 years later
- Scalable performance, simple explanation
- What is NewSQL
- Overview about NoSQL databases
- Performance loss in OLTP systems
- Memory price trends
- (wiki) Shared Nothing Architecture
- (wiki) Column oriented DBMS
- How NewSQL handles big data
- What is YCSB benchmark
- What is TPC benchmark
- Transactional isolation levels

Links: NuoDB

- http://www.infoq.com/articles/nuodb-architecture-1/
- http://www.infoq.com/articles/nuodb-architecture-2/
- http://stackoverflow.com/questions/14552091/nuodb-and-hdfs-as-storage
- http://go.nuodb.com/rs/nuodb/images/NuoDB_Benchmark_Report.pdf
- NuoDB white paper (google has you :)
- https://aphyr.com/posts/292-call-me-maybe-nuodb
- http://dev.nuodb.com/techblog/failure-detection-and-network-partition-management-nuodb

Links: VoltDB

- White paper, Technical overview (google has you)
- https://github.com/VoltDB/voltdb-client-erlang/blob/master/doc/BENCHMARK1.md
- http://www.mysqlperformanceblog.com/2011/02/28/is-voltdb-really-as-scalable-as-they-claim/
- https://voltdb.com/blog/voltdb-3-x-performance-characteristics/
- http://docs.voltdb.com/UsingVoltDB/KsafeNetPart.php
- https://news.ycombinator.com/item?id=6639127

Links: ScaleDB

- http://scaledb.com/pdfs/TechnicalOverview.pdf
- http://www.scaledb.com/pdfs/scaledb_multitenant.pdf
- http://www.percona.com/live/mysql-conference-2013/sites/default/files/slides/DB_Vistualization_for_PublicPrivate_Clouds.pdf

Links: Clustrix

- http://www.clustrix.com/wp-content/uploads/2013/10/Clustrix_A-New-Approach_WhitePaper.pdf
- http://www.clustrix.com/wp-content/uploads/2013/10/Clustrix_Driving-the-New-Wave_WP.pdf
- http://www.clustrix.com/wp-content/uploads/2013/10/Clustrix_AWS_WP.pdf
- http://www.clustrix.com/wp-content/uploads/2013/10/Clustrix_TPCC_Percona.pdf
- http://sergei.clustrix.com/2011/01/mongodb-vs-clustrix-comparison-part-1.html
- http://docs.clustrix.com/display/CLXDOC/Consistency%2C+Fault+Tolerance%2C+and+Availability

Links: FoundationDB

- https://foundationdb.com/key-value-store/white-papers
- http://blog.foundationdb.com/call-me-maybe-foundationdb-vs-jepsen
- https://foundationdb.com/acid-claims
- https://foundationdb.com/key-value-store/performance
- https://foundationdb.com/layers/sql/documentation/Concepts
- https://foundationdb.com/layers/sql/documentation/SQL/indexes.html
- https://foundationdb.com/layers/sql/performance
- https://foundationdb.com/key-value-store/features
- https://foundationdb.com/key-value-store/documentation/configuration.html
- https://foundationdb.com/key-value-store/documentation/beta1/developer-guide.html
- https://foundationdb.com/layers/sql/documentation/Concepts/known.limitations.html

Links: MemSQL

- MemSQL Whitepaper "The Modern Database Landscape"
- MemSQL Whitepaper "ESG Lab Benchmark of MemSQL's Performance”
- MemSQL Whitepaper “Technical overview”
- http://developers.memsql.com/docs/latest/concepts/dev_concepts.html
- http://developers.memsql.com/docs/2.6/admin/high_availability.html