Scalable SQL and NoSQL Data Stores

Rick Cattell

Presenter: MoHan Zhang

What is NoSQL?

NoSQL

- Stand for: Not Only SQL / Not Relational
- Features:
  - Ability to scale to many servers
  - Efficient use of distributed indexes & RAM for data storage
  - Dynamically add new attributes to data records (dynamic schema)
  - Weaker concurrency model than ACID transactions of most relational databases

ACID vs BASE

- ACID: Atomicity, Consistency, Isolation, Durability
- BASE: Basically Available, Soft State, Eventually Consistent
  - Updates are eventually propagated, but limited guarantee on read consistency
- Give up ACID constraints = Higher Performance and Scalability

Key Property: Shared Nothing Architecture

- Replicate and partition data over many servers
    - support a large number of simple read/write operations per second

The purpose of this paper is to survey a set of **scalable** SQL and NoSQL database models under the following 4 categories:

- Key-value Stores
- Document Stores
- Extensible Record Stores
- Relational Databases

- Key-value Stores
- Document Stores
- Extensible Record Stores
- Relational Databases

Key-value Stores

- Systems under this category store values and an index to find them, based on a programmer defined key
- Insert, Delete, Lookup Operations
- Scalability through key distributions over nodes

Use Case:

- Simple application, one kind of object, only need to look up on one attribute

Project Voldemort

A distributed database.

Project Voldemort

- Written in Java, open-source, supported by Linkedin
- Multi-version Concurrency Control (MVCC) for updates
    - No guarantee of consistent data
- Optimistic Locking
- Consistent Hashing
- Store data in RAM or in storage engines

riak

Riak

- Written in Erlang, open-source, client based on RESTful
- Objects can be fetched and stored in JSON
    - can have multiple fields (like documents)
- Only lookup is on Primary Key
- MVCC & Consistent Hashing
- Map/Reduce to split work over nodes in a cluster
- Unique Feature: Store links between objects

redis

Redis

- Written in C, Open-source
- Client side does the distributed hashing over servers, servers store data in RAM
- Updates by locking
- Asynchronous Replication

membase

Membase

- Based on distributed in-memory indexing system, Memcache
- Open-source
- Elastically add / remove servers in a running system

Other systems:

- Scalaris
- Tokyo Cabinet

	Riak	Redis	Scalaris	Tokyo Cabinet	Membase	Voldemort
Data Store	Ram or disk	Ram	Ram	Ram or disk	Ram	Ram or disk
Replicatio n	Async	Async	Sync	Async	Sync	Async
Transactio ns	No	No	Yes	Yes	No	No
Updates	MVCC	Locking	Locking	Locking	Locking	MVCC

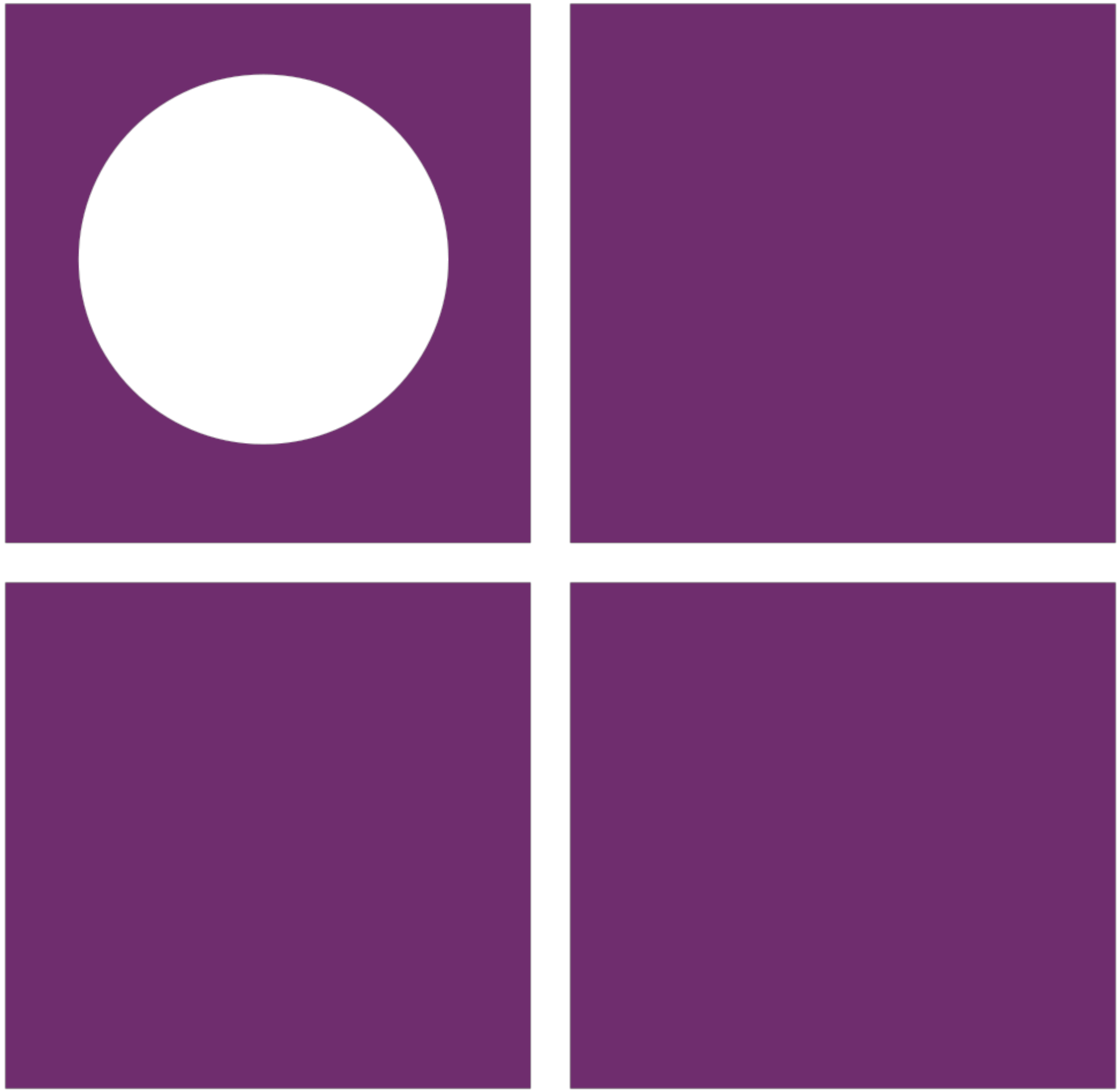
- Key-value Stores
- Document Stores
- Extensible Record Stores
- Relational Databases

Document Stores

- Systems under this category store documents. Documents are indexed and a query mechanism is provided.
- Secondary indexes and multiple types of objects per database
- No ACID Transactional Properties

Use Case:

- Multiple kinds of objects (e.g. Driver Licensing, with vehicles and drivers), need to look up on multiple attributes (driver_name, license_number, owned_vehicle, birthday)
- Need to tolerate eventual consistency



SimpleDB

- Pay as you go service from Amazon
- Select, Delete, GetAttributes, PutAttributes
- Does not allow nested documents
- Eventual Consistency & Async replication
- More than one grouping in one database
    - multiple indexes
- No automatic data partitioning over servers

mongoDB®

MongoDB

- Written in C++, GPL Open-source
- Automatic sharing distributed documents over many servers
- Replication used for failover, not for scalability
- Data stored in BSON format (binary JSON)
- Master-slave replication with automatic failover and recovery

Other systems

- CouchDB
- Terrastore

- Key-value Stores
- Document Stores
- Extensible Record Stores
- Relational Databases

Extensible Record Stores

- Systems under this category store extensible records that can be partitioned vertically and horizontally across nodes
- Motivated by Google's BigTable, but none achieved the scalability of BigTable

Use Case:

- Multiple kinds of objects and need to look up on multiple attributes, higher throughput than Document Stores, stronger concurrency
- e.g. eBay application:
  - cluster users by country
  - Separate rarely changed customer information in one place, and frequently updated information in another place for improvements in performance

APACHE
HBASE

HBase

- Written in Java, Apache project
- Hadoop DFS, updates in memory and periodically write to disk
- updates go to the end of data files
- B-trees allow fast range queries and sorting
- Optimistic Concurrency control

HYPERTABLE INC

Hypertable

- Written in C++, Open-source, sponsored by Baidu
- Similar to BigTable and HBase
- Uses query language named HQL

cassandra

Cassandra

- Written in Java, Open-source, basic features similar to HBase
- Used by Facebook and other companies
- Weaker Concurrency Model: No locking, Async replica updates

- Key-value Stores
- Document Stores
- Extensible Record Stores
- Relational Databases

Scalable Relational Databases

- Pre-defined Schema, SQL interface, ACID transactions
- Penalize Large-scope operations, while NoSQL systems forbid these operations
- Avoid cross-node operations to deliver scalability

Use Case:

- Many tables across different kinds of data, need for a centralized schema, need for simplicity of SQL
- Database being updated from many locations

MySQL™ Cluster

MySQL Cluster

- Shared nothing architecture: shards data over multiple database servers
- In-memory & Disk-based data
- Can scale to more nodes than other RDBMSs but runs into bottleneck after a few dozen nodes

VOLTDB

VoltDB

- Open-source RDBMS, designed for scalability and per-node performance
- Tables partitioned over many servers
- Shards replicated for crash recovery
- Designed for databases that fit into distributed RAM of a server, so that the system never waits for the disk
    - This and other optimizations boost single node performance

Clustrix

Clustrix

- Nodes sold as rack-mounted appliances
- Scalability to hundreds of nodes, automatic sharing & replication
- Automatic failover and failure recovery
- Seamlessly compatible with MySQL

Other systems

- ScaleDB
- ScaleBase
- NimbusDB

Conclusion

Some predictions from 2010

- Many developers are willing to abandon globally ACID transactions in order to gain scalability, availability, and other advantages
- The simplicity, flexibility, and scalability of NoSQL data stores fill a niche market
- Many data models described today will not be enterprise ready in a while
- One or two systems within each category will become the leader

Relational > NoSQL?

- Relational can do everything NoSQL can, with analogous performance and scalability, adding in the convenience of SQL
- Relational DBMSs have been dominating the market for more than 30 years
- Relational DBMSs have been built to deal with other problems and they will have no problem dealing with scalability

NoSQL > Relational?

- No benchmarks showing Relational can achieve the scalability of some NoSQL systems
- In NoSQL: only pay the learning curve for the complexity you require
- Relational DBMS makes expensive (multi-node, multi-table) operations too accessible, NoSQL systems make them impossible or visibly expensive to programmers
- While relational DBMSs have been successful, over the years there have been other products occupying niche markets

Thank you!

Q&A

System	Conc Contol	Data Storage	Repli-cation	Tx
Redis	Locks	RAM	Async	N
Scalaris	Locks	RAM	Sync	L
Tokyo	Locks	RAM or disk	Async	L
Voldemort	MVCC	RAM or BDB	Async	N
Riak	MVCC	Plug-in	Async	N
Membrain	Locks	Flash + Disk	Sync	L
Membase	Locks	Disk	Sync	L
Dynamo	MVCC	Plug-in	Async	N
SimpleDB	None	S3	Async	N
MongoDB	Locks	Disk	Async	N
Couch DB	MVCC	Disk	Async	N
Terrastore	Locks	RAM+	Sync	L
HBase	Locks	Hadoop	Async	L
HyperTable	Locks	Files	Sync	L
Cassandra	MVCC	Disk	Async	L
BigTable	Locks+stamps	GFS	Sync+ Async	L
PNUTs	MVCC	Disk	Async	L
MySQL Cluster	ACID	Disk	Sync	Y
VoltDB	ACID, no lock	RAM	Sync	Y
Clustrix	ACID, no lock	Disk	Sync	Y
ScaleDB	ACID	Disk	Sync	Y
ScaleBase	ACID	Disk	Async	Y
NimbusDB	ACID, no lock	Disk	Sync	Y