SCALABLE SQL & NOSQL DATA STORES

TABLE OF CONTENTS

SECTION 1

OVERVIEW & SCOPE

KNOW THE AUTHOR

- Independent Database Consultant
- 20+ years in Sun Microsystems
- Proficient in Database Scalability, Enterprise Java, Object oriented databases
- Co founder of SQL Access and co-creator of JDBC

6 KEY FEATURES OF NO-SQL

- Ability to horizontally scale “simple operations” throughput over many servers.
- The ability to replicate and to distribute data over many servers
- A simple call level interface or protocol
- A weaker concurrency model than the ACID transactions
- Efficient use of distributed indexes and RAM for data storage
- The ability to dynamically add new attributes to data records.

SCOPE AND JARGONS

- Simple Operations – Key lookups, reads and writes of one record or a small number of records.
- Horizontal Scalability – Ability to distribute data and the load of Simple Operations to many servers.
- Horizontal Scalability is different from Vertical Scalability.
- Horizontal and Vertical Partitioning is not related to scaling
- Will realize the various data stores available and compare them

BEYOND SCOPE

- Graph Database Systems
- Object-oriented Database Systems
- Distributed object-oriented stores

DATA MODEL TERMINOLOGIES

- TUPLE – a row in relational table
- DOCUMENT – values to be nested documents or lists as well as scalar values and attribute names are dynamic
- EXTENSIBLE RECORD – Hybrid between Tuple and Document
- OBJECT – Analogous to objects in programming languages

DATA STORE CATEGORIES

- KEY VALUE STORES
- DOCUMENT STORES
- EXTENSIBLE RECORD STORES
- RELATIONAL DATABASES

SECTION 2

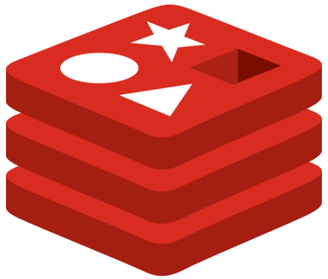
MOSTLY NO-SQL

PROJECT VOLDEMORT

- Written in Java and provides Multi-version Concurrency Control (MVCC)
- Updates Replicas Asynchronously
- No guaranteed consistency
- Optimistic Locking – If updates conflict with any other process, they can be backed out
- Automatic Sharding of Data
- Automatic recovery of failed nodes

- Written in Erlang
- Key Value and a Document Store
- Lacks important features of document stores
- Can be stored in JSON, objects can be grouped into Buckets
- Supports indices only on primary key and not on other fields
- Riak allows replication of objects and sharding by hashing on primary key.
- Self repairs out-of-sync updates
- Can store links between objects
- Client Interface is based on RESTful HTTP

```
{
 "bucket":"customers",
 "key":"12345",
 "object":{
          "name":"Mr. Smith",
          "phone":”415-555-6524”  }
 "links":[
         ["sales","Mr. Salesguy","salesrep"],
         ["cust-orders","12345","orders"] ]
 "vclock":"opaque-riak-vclock",
 "lastmod":"Mon, 03 Aug 2009 18:49:42 GMT"
}
```

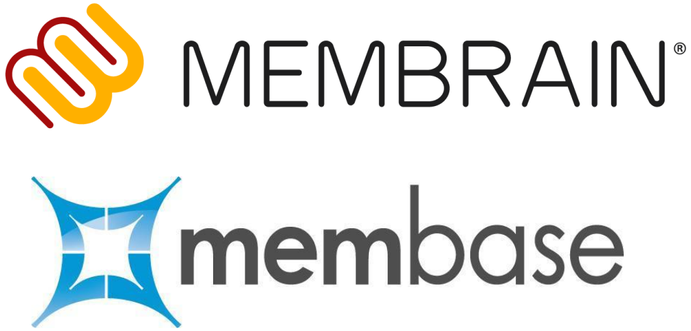


redis

- Written in C
- Insert, Update and Delete Operations
- Includes List and Set Operations
- Data is stored in RAM
- Data can be copied to disk for backup
- Automatic Updates
- Asynchronous Replication

- Written in Erlang
- Key Ranges are distributed over nodes
- Better Load Balancing
- Insert, Update and Delete Operations
- Multi-node power failure will cause disastrous data loss

- Written in C
- 6 different variations
- Hash indexes in memory or on disk
- B-Trees in memory or on disk
- Fixed-size record tables
- Variable length record tables
- Claims to support locking, ACID and binary array data type
- Asynchronous Replication

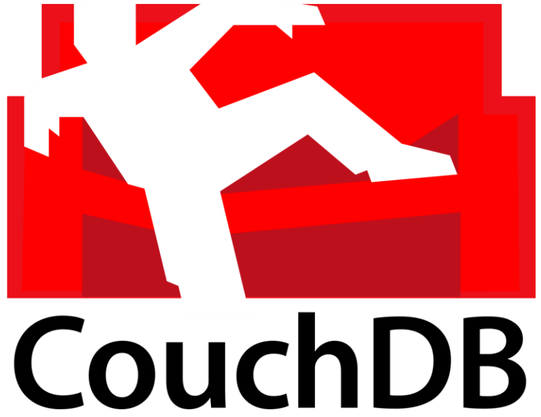


- Membase and Membrain supports persistence and replication
- Membase – Ability to elastically add or remove servers in a running system.
- Membrain – Excellent tuning of flash memory

KEY STORE SUMMARY

- All support Insert Update and Delete
- Voldemort, Riak, Tokyo Cabinet and Memcached systems can store data in RAM with storage add ons.
- Scalaris & Memcached - Synchronous replication | Scalaris & Tokyo Cabinet – Transactions
- Voldemort & Riak use MVCC

- Has Select, Delete, GetAttributes and PutAttributes operations on documents
- Does not allow nested documents
- Eventual Consistency
- Supports more than one grouping in one database
- Fixed-size record tables
- Documents are put into domain that support multiple indexes
- Does not automatically partition data over servers
- Asynchronous Replication



- A CouchDB is similar to SimpleDB
- Queries are done in CouchDB called as “views”
- B-tree indexes
- Queries can be distributed over many nodes in parallel
- Asynchronous replication
- Does not guarantee consistency
- Durability on System crash
- MVCC

- Written in C++
- Provides automatic sharding
- Replication only for failovers and not scalability
- Supports dynamic queries
- Provides atomic operations on a table - findAndModify
- ‘update if current’ for changing a document only if field matches given value
- Supports Map Reduce
- Stores data in binary-like JSON format called BSON
- Asynchronous Replication
- Automatic Failover and Recovery

“To find all products released last year costing under $100 you could write”

```
db.products.find(
  {released: {$gte: new Date(2009, 1, 1,)},
  price {‘$lte’: 100},})
```

- Terrastore can automatically partition data over server nodes
- Can automatically redistribute data when servers are added or removed
- Like MongoDb, it can query values with ranges
- Like CouchDb, it includes map/reduce mechanism
- Schema-less
- Supports Replication

DOCUMENT STORE SUMMARY

- Schema-less
- SimpleDB Domain = CouchDB Database = MongoDB Collection =Terrastore Bucket.
- SimpleDB calls documents as “items”, CouchDB calls an attribute as fields
- Can query collections based on multiple attribute value constraints

EXTENSIBLE RECORD STORES

- Rows are split across through sharding on primary key
- Columns are distributed over multiple nodes by column-groups
- Both can be used for same table
- Most of the stores were patterned after Google’s Big Table

- Written in Java and uses HDFS
- Row operations are atomic with row-level locking
- Optimistic Concurrency Control
- Multiple Master Support
- Fixed-size record tables
- Map/Reduce support

HYPERTABLE INC

- Written in C++ and similar to Hbase and Big Table
- Tables are replicated over servers by key ranges

- Has a weaker concurrency model
- Failure detection and recovery are automatic
- “Supercolumns” provide another level of grouping within column groups
- Any row can have any combination of column values
- Has an ordered hash index

SECTION 3

SCALABLE RELATIONAL SYSTEMS

RDBMS SCALABILITY

SMALL SCOPE OPERATIONS

SMALL SCOPE TRANSACTIONS

- Should not span many nodes
- Should not join many tables
- Should penalize the user if large scale operation is used
- SQL and ACID will make it advantageous over NoSql

- Works by replacing the InnoDB engine with a distributed layer called NDB.
- Shards data over multiple database servers
- Every shard is replicated
- In-memory as well disk based storage
- Runs into bottleneck after a few dozens of nodes

- Tables are partitioned over many servers and clients can call any server.
- Distribution is transparent to users
- Selected tables can be replicated for fast-access to read mostly data
- Schema changes are currently limited
- Eliminates “waits in SQL” execution.

- Similar to VoltDB and MYSQL Cluster
- Claims scalability to 100s of nodes
- Automatic Sharding and Replication
- Failover and fail node recovery is automatic
- Load balancing is transparent

- Horizontal Scaling on top of MySql Cluster
- Shards tables over multiple single-node MySQL databases

- Replaces the InnoDB of the MySQL Cluster
- Clustering of multiple servers to achieve scalability
- Architecture not very scalable
- Automatic Sharding and row level locking

- MVCC and distributed object storage
- SQL and AVL tree indexes
- Horizontal segmentation of data

KEY VALUE STORE USE CASE

- Simple Application with one kind of object
- More look-ups and rare update of data
- Looked up using single attribute
- Should move to a document store when records are looked up using multiple attributes

DOCUMENT STORE USE CASE

- Application with multiple and different kind of objects
- Department of Motor Vehicles with vehicles and drivers
- Eventual consistency
- Limited atomicity
- Isolation

EXTENSIBLE RECORD STORE USE CASE

- Application with multiple and different kind of objects
- Higher throughput and stronger concurrency guarantees
- E-Bay Style application
- Vertical and Horizontal Partitioning

SCALABLE RDBMS USE CASE

- If application requires many tables with different types of data
- SQL is always easy and better than lower level commands
- ACID guarantees
- Many tools are available
- Scalable RDBMS are improving

ACID TRADE OFF

NO SQL ADVANTAGE

THE SCALABLE STORE DELAY

FUTURE PREDICTIONS

CONSOLIDATION OF SYSTEMS

NEO RDBMS

SYSTEM COMPARISONS

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

THANK YOU!

ANY QUESTIONS?