

MongoDB and CouchDB

The
Pragmatic
Programmers

Instructor: Dr. GP Saggese - gsaggese@umd.edu

- All concepts in slides
- MongoDB tutorial
- Web
 - <https://www.mongodb.com/>
 - Official docs
 - pymongo
- Book
 - Seven Databases in Seven Weeks, 2e

Seven Databases in Seven Weeks

Second Edition

A Guide to Modern
Databases and the
NoSQL Movement

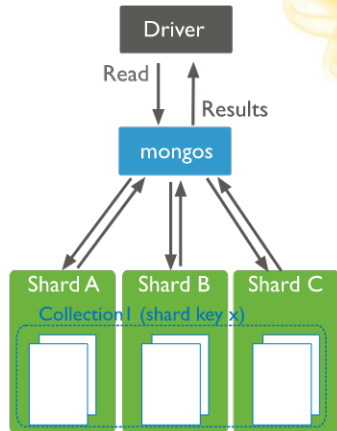
Luc Perkins
with Eric Redmond and Jim R. Wilson

Series editor: Bruce A. Tate
Development editor: Jacquelyn Carter



MongoDB Processes and Configuration

- **mongod**: database instance (i.e., a server process)
- **mongosh**: interactive shell (i.e., a client)
 - Fully functional JavaScript environment for use with a MongoDB
- **mongos**: database router
 - Process all requests
 - Decide how many and which **mongod** instances should receive the query (sharding / partitioning)
 - Collate the results
 - Send result back to the client
- You should have:
 - One **mongos** (router) for the whole system no matter how many **mongods** you have; or
 - One local **mongos** for every client if you wanted to minimize network latency

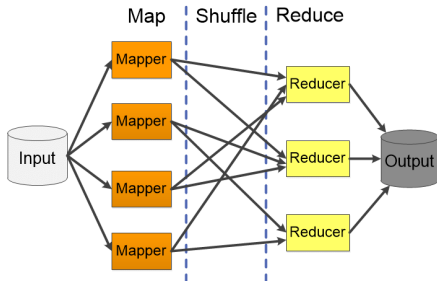
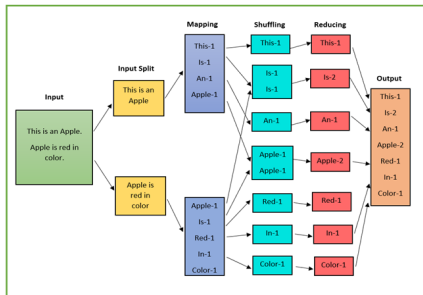


MapReduce Functionality

- Perform map-reduce computation given a collection of (keys, value) pairs
- Must provide at least a map function, reduction function, and the name of the result set

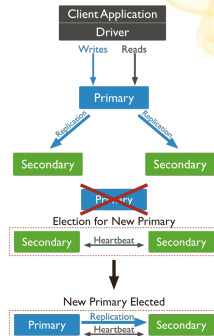
```
db.collection.mapReduce(
  <map\_function>,
  <reduce\_function>,
  {
```

```
    out: <collection>,
    query: <document>,
    sort: <document>,
    limit: <number>,
    finalize: <function>,
    scope: <document>,
    jsMode: <boolean>,
    verbose: <boolean>
```



Data Replication

- **Data replication** ensure:
 - Redundancy
 - Backup
 - Automatic failover
- Replication occurs through groups of servers known as **replica sets**
 - **Primary set**: set of servers that client asks direct updates to
 - **Secondary set**: set of servers used for duplication of data
 - Different properties can be associated with a secondary set,
 - E.g., secondary-only, hidden delayed, arbiters, non-voting
- If the primary fails the secondary sets “vote” to elect the new primary set



Sync vs Async Replication

- **Synchronous replication:**

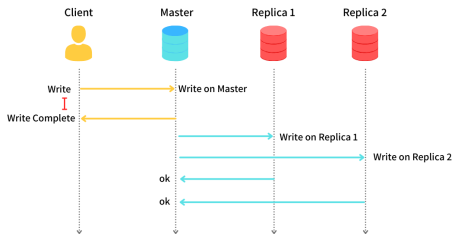
updates are propagated to other replicas as part of a single transaction

- Implementations

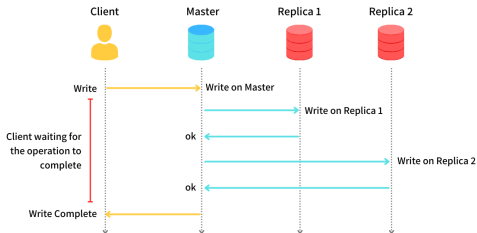
- 2-Phase Commit (2PC)
- Paxos
- Both solutions are complex / expensive

- **Asynchronous replication**

- The primary node propagates updates to replicas
- The transaction is completed before replicas are updated (even if there are failures)
- Commits are quick at cost of consistency



Asynchronous Replication



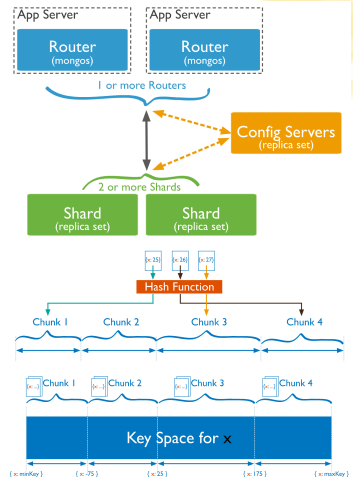
Synchronous Replication

Data Consistency

- **Client decides how to enforce consistency for reads**
- Reads to a primary have **strict consistency**
 - Reads reflect the latest changes to the data
 - All writes and *consistent* reads go to the primary
- Reads to a secondary have **eventual consistency**
 - Updates propagate gradually
 - Client may read a previous state of the database
 - All *eventually consistent* reads are distributed among the secondaries

MongoDB: Sharding

- **Shard** = subset of data
 - A collection is split in pieces based on the shard key
 - Data distributed based on shard key or intervals [a, b)
- **Sharding** = method for distributing data across different machines
- **Horizontal scaling** can be achieved through sharding
 - Divide data and workload over multiple servers
 - Complexity in infrastructure and maintenance
- **mongos** acts as a query router interfacing clients and sharded cluster
 - Each shard can be deployed as a replica set
 - Config servers store metadata and configuration settings for cluster



RDMBS Internals

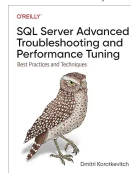
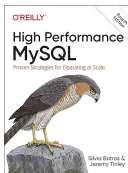
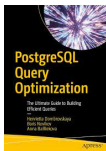
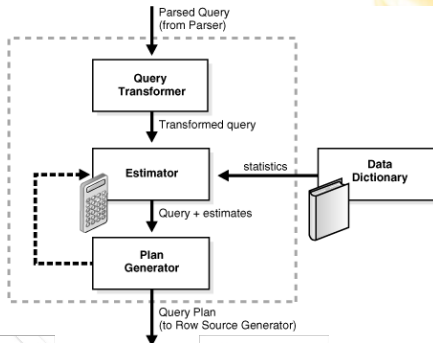
Storage hierarchy - How are tables mapped to files? - How are tuples mapped to disk blocks?

Buffer Manager - Bring pages from disk to memory - Manage the limited memory

Query Processing Engine - Given a user query, decide how to “execute” it - Specify sequence of pages to be brought in memory - Operate upon the tuples to produce results

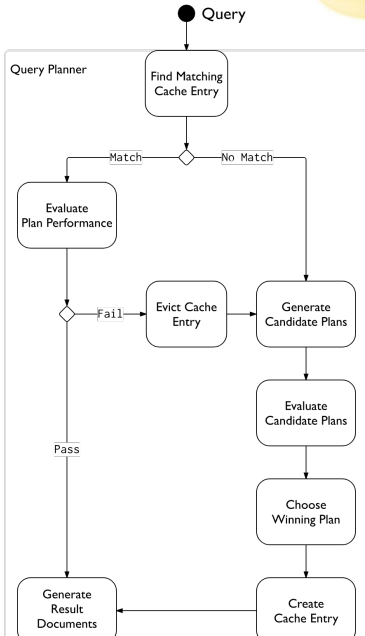
Query Optimizer

- **RDBMSs: query optimizer is static**
 - Assign a cost to each query plan
 - Estimate some cost params (e.g., time to access data)
 - Search for the best query
 - At least traditional RDBMSs



Query Optimizer

- **MongoDB: query optimizer is dynamic**
 - Try different query plans and learn which ones perform well
 - The space of query plans is not so large, because there are no joins
 - When testing new plans
 - Execute multiple query plans in parallel
 - As soon as one plan finishes, terminate the other plans
 - Cache the result
 - If a plan that was working well starts performing poorly try again different plans
 - E.g, data in the DB has changed, parameter values to a query are different



MongoDB: Strengths

- Provide a flexible and modern query language
- High-performance
 - Implemented in C++
- Very rapid development, open source
 - Support for many platforms
 - Many language drivers
- Built to address a distributed database system
 - Sharding
 - Replica sets of data
- Tunable consistency
- Useful for working with a huge quantity of data not requiring a relational model
 - The relationships between the elements does not matter
 - What matters is the ability to store and retrieve great quantities of data

MongoDB: Limitations

- No referential integrity
 - Aka foreign key constraint
- Lack of transactions and joins
- High degree of denormalization
 - Need to update data in many places instead of one
- Lack of predefined schema is a double-edged sword
 - You must have a data model in your application
 - Objects within a collection can be completely inconsistent in their fields
- CAP Theorem: targets consistency and partition tolerance, giving up on availability

