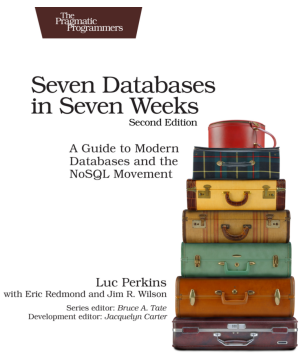
6.3: MongoDB Config

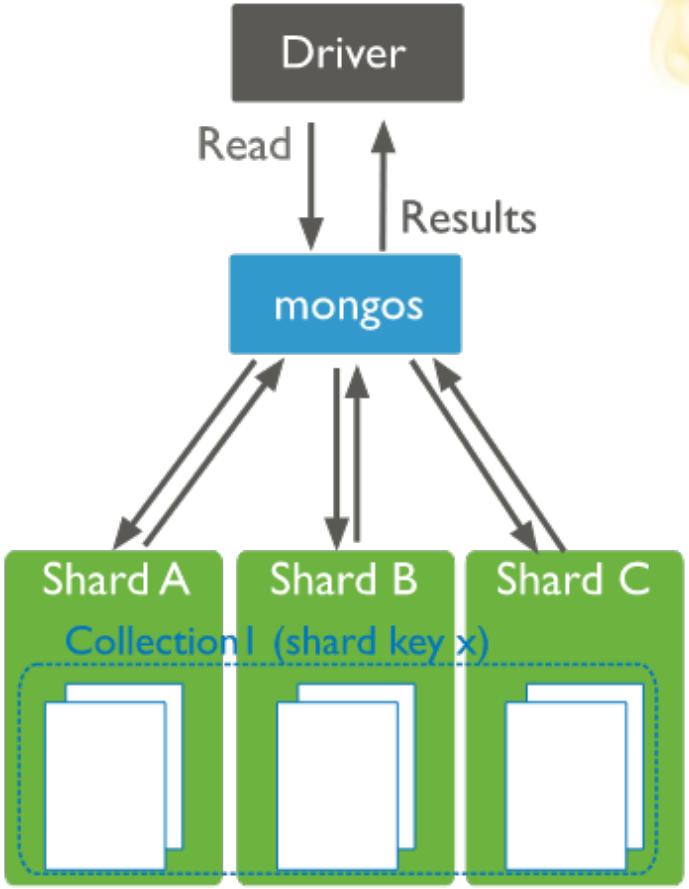


Instructor: Dr. GP Saggese - gsaggese@umd.edu

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

MongoDB Processes and Configuration

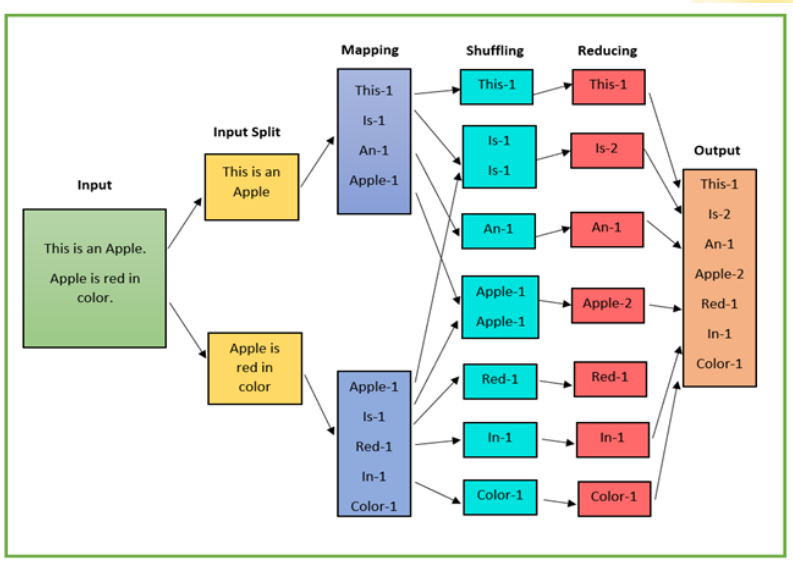
- `mongod`: database instance (server process)
- `mongosh`: interactive shell (client)
  - JavaScript environment for MongoDB
- `mongos`: database router
  - Process requests
  - Decide which `mongod` instances receive the query (sharding/partitioning)
  - Collate results
  - Send result to client
- You should have:
  - One `mongos` (router) for the system regardless of `mongod` count; or
  - One local `mongos` per client to minimize network latency

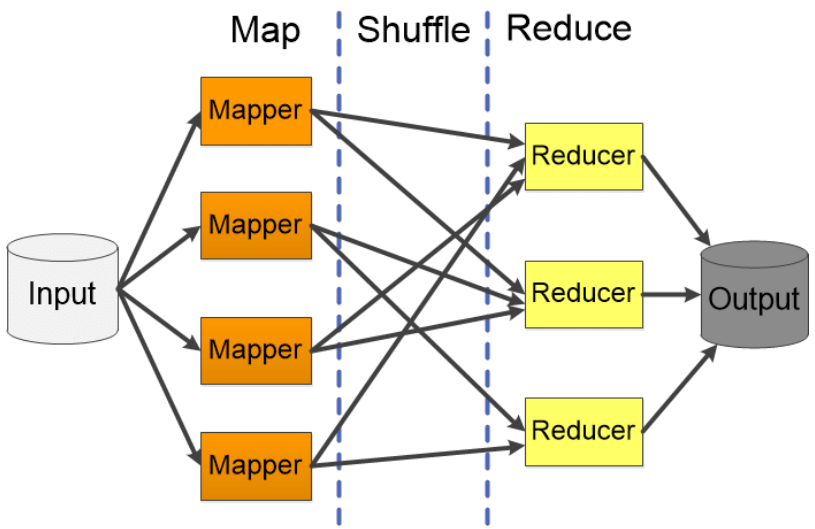


MapReduce Functionality

- Perform map-reduce computation on (key, value) pairs
- Provide map function, reduction function, and result set name

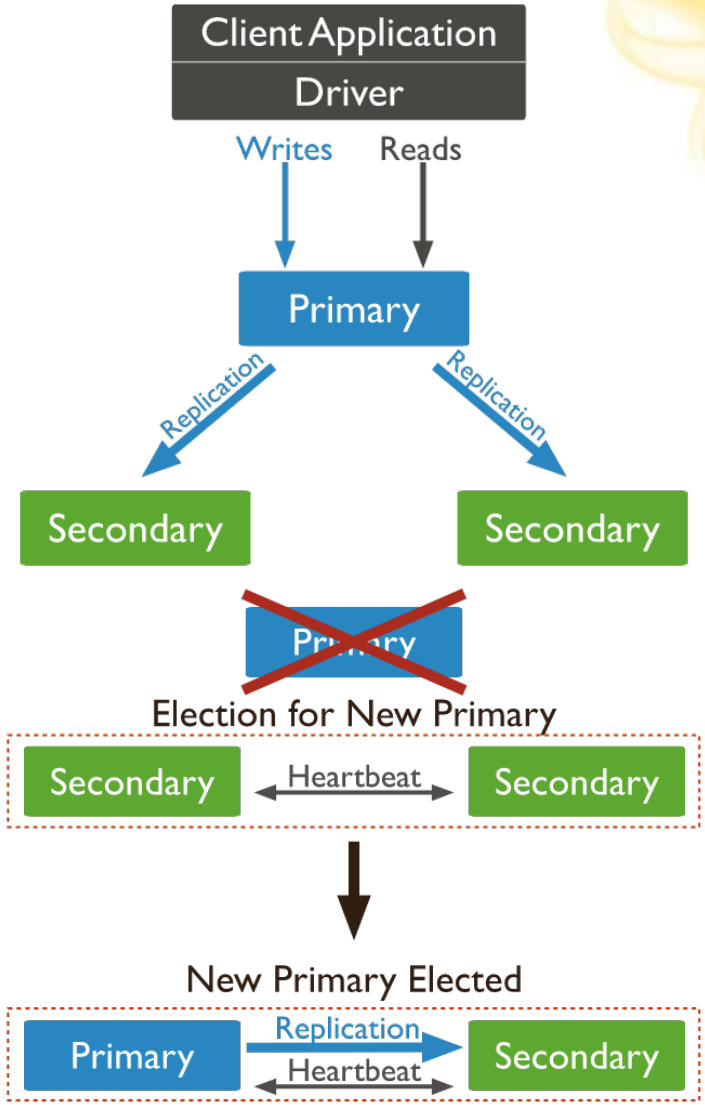
```
db.collection.mapReduce(
  <map_function>,
  <reduce_function>,
  {
    out: <collection>,
    query: <document>,
    sort: <document>,
    limit: <number>,
    finalize: <function>,
    scope: <document>,
    jsMode: <boolean>,
    verbose: <boolean>
  })
```





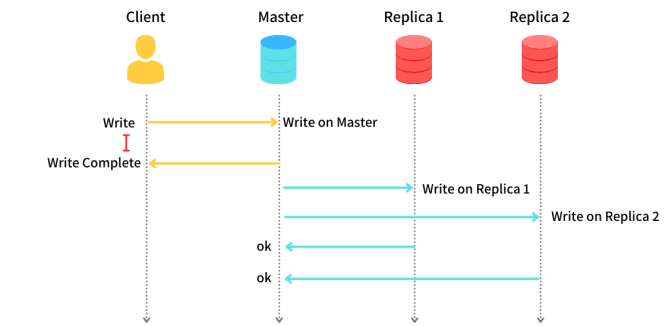
Data Replication

- **Data replication** ensure:
  - Redundancy
  - Backup
  - Automatic failover
- Replication occurs through groups of servers known as **replica sets**
  - **Primary set**: servers for direct updates
  - **Secondary set**: servers for data duplication
  - Properties of secondary set:
    - Secondary-only, hidden, delayed, arbiters, non-voting
- If primary fails, secondary sets "vote" to elect new primary

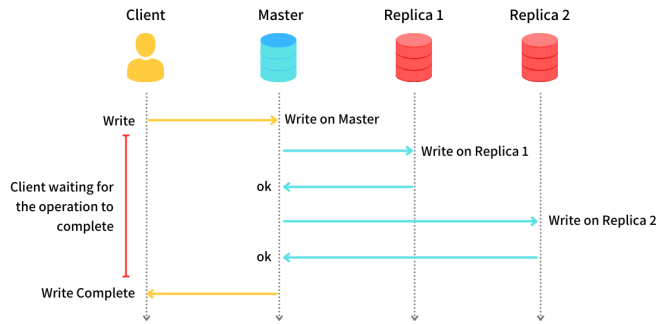


Sync vs Async Replication

- **Synchronous replication**: updates propagate to replicas in a single transaction
- Implementations
  - 2-Phase Commit (2PC)
  - Paxos
  - Complex and expensive
- **Asynchronous replication**
  - Primary node propagates updates to replicas
  - Transaction completes before replicas update (even if failures occur)
  - Quick commits at consistency cost



Asynchronous Replication



Synchronous Replication

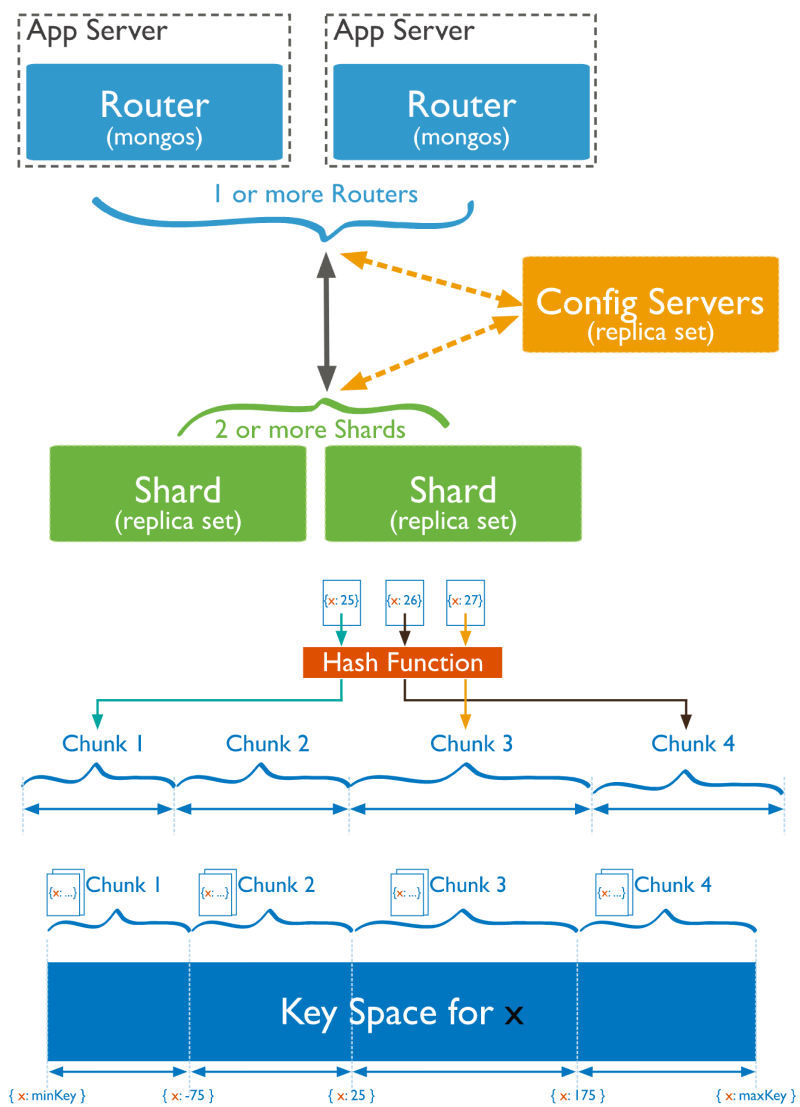
SCIENCE ACADEMY

Data Consistency

- **Client decides how to enforce consistency for reads**
- Reads to a primary have **strict consistency**
  - Reflect latest data changes
  - All writes and consistent reads go to primary
- Reads to a secondary have **eventual consistency**
  - Updates propagate gradually
  - May read previous database state
  - Eventually consistent reads distributed among secondaries

MongoDB: Sharding

- **Shard** = subset of data
  - Split collection based on shard key
  - Distribute data based on shard key or intervals [a, b)
- **Sharding** = method for distributing data across machines
- **Horizontal scaling** achieved through sharding
  - Divide data and workload over servers
  - Complexity in infrastructure and maintenance
- `mongos` acts as query router interfacing clients and sharded cluster
  - Deploy each shard as a replica set
  - Config servers store metadata and configuration settings for cluster

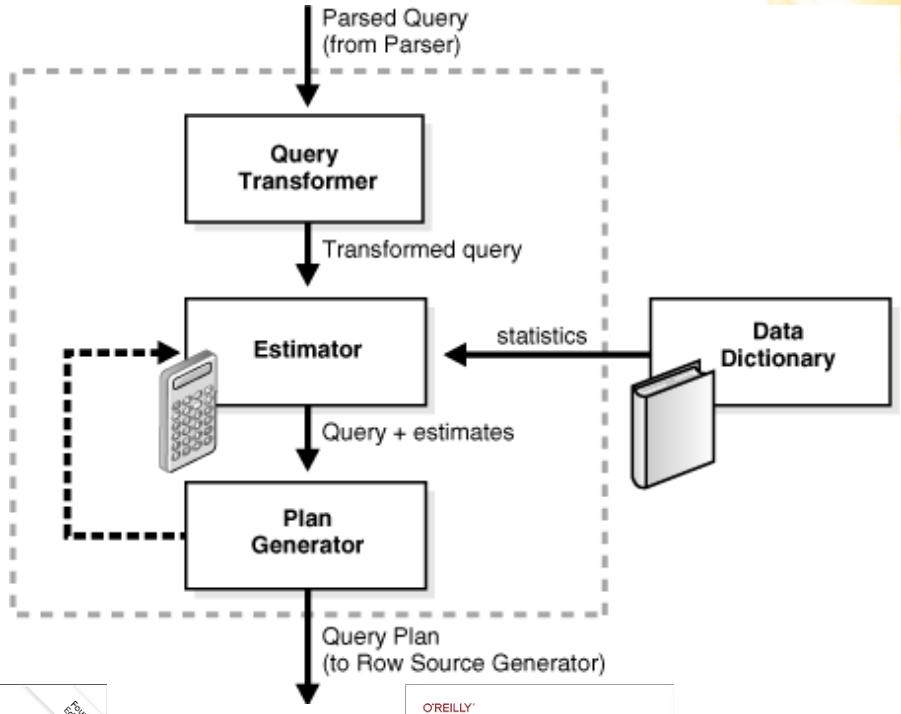


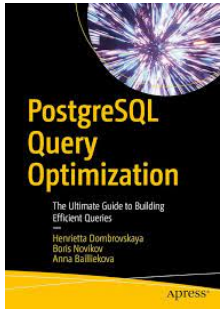
RDBMS Internals

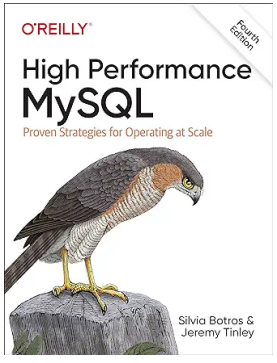
- **Storage hierarchy**
  - Map tables to files
  - Map tuples to disk blocks
- **Buffer Manager**
  - Bring pages from disk to memory
  - Manage limited memory
- **Query Processing Engine**
  - Execute user query
  - Specify sequence of pages for memory
  - Operate on tuples to produce results

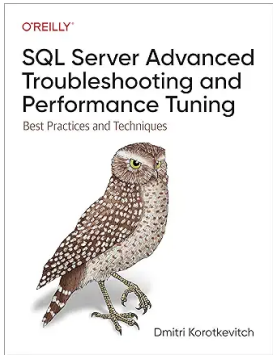
Query Optimizer

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



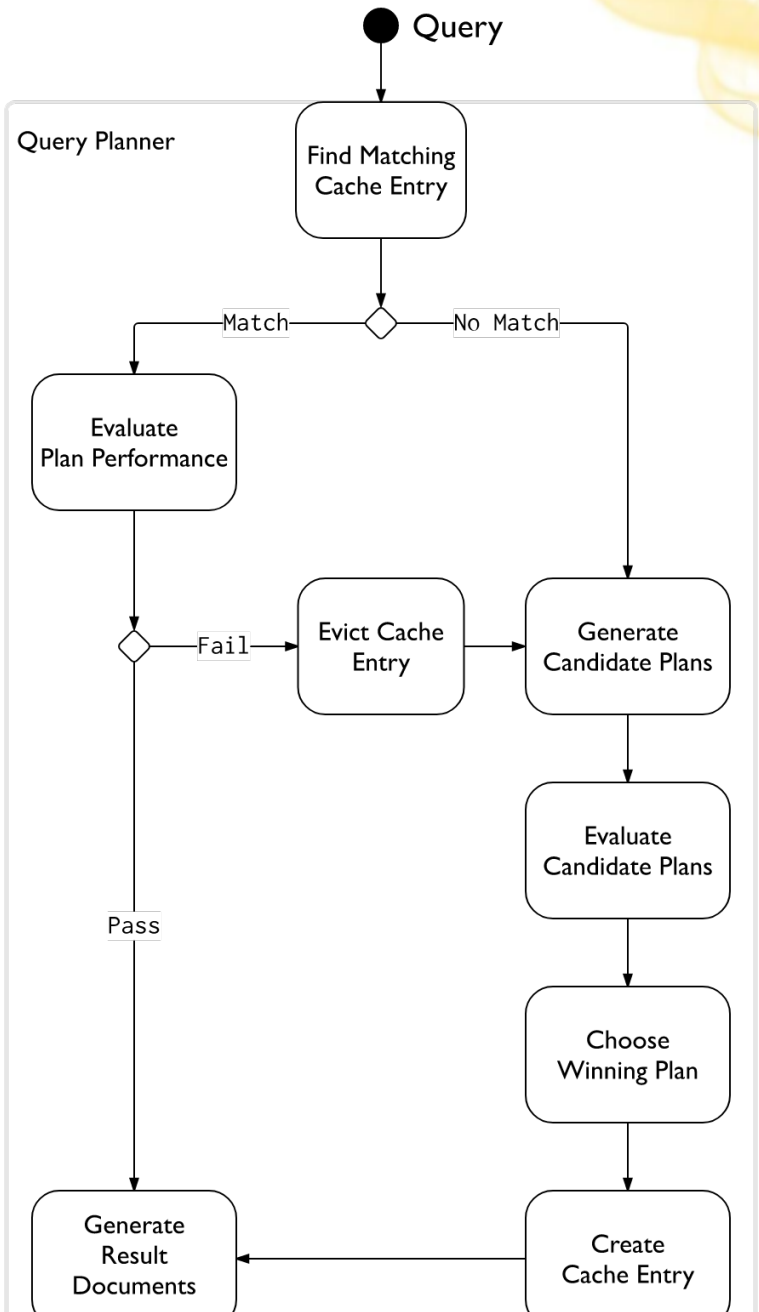






Query Optimizer

- **MongoDB: query optimizer is dynamic**
  - Try different query plans, learn which perform well
  - Query plan space is small, no joins
  - Testing new plans
    - Execute multiple plans in parallel
    - Terminate others when one finishes
  - Cache result
  - If a plan performs poorly, try different plans
    - E.g., data changed, query parameters differ



MongoDB: Strengths

- Provide flexible, modern query language
- High-performance
  - Implemented in C++
- Rapid development, open source
  - Supports many platforms
  - Multiple language drivers
- Built for distributed database systems
  - Sharding
  - Replica sets
- Tunable consistency
- Ideal for large data not needing relational model
  - Element relationships irrelevant
  - Focus on storing and retrieving large data quantities

MongoDB: Limitations

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