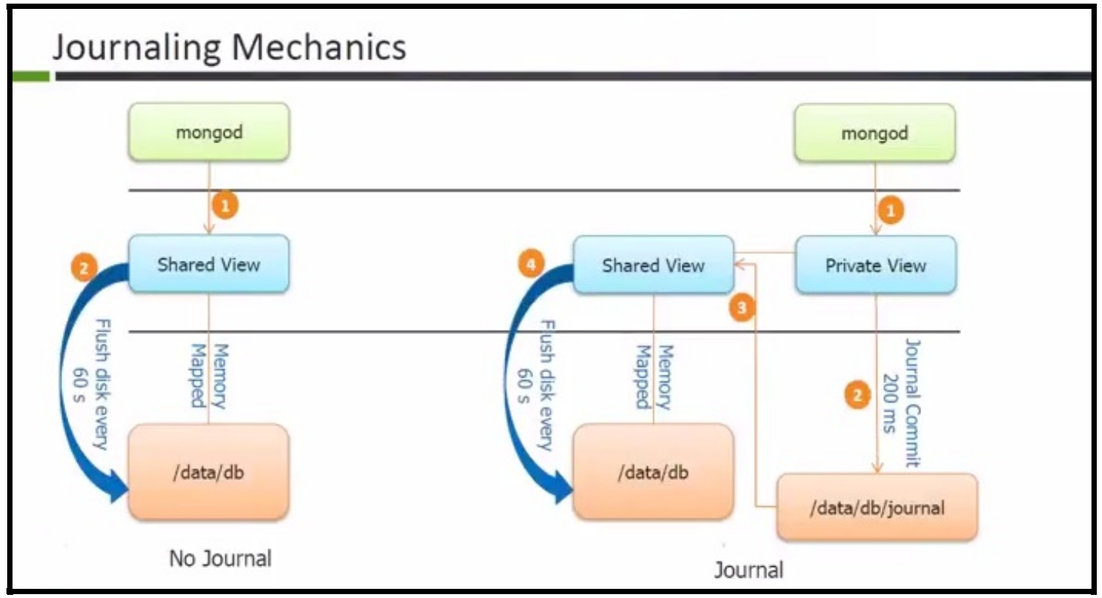
MongoDB Replication

1. **MongoDB write path**
2. Replication principles
3. Replica set Read and Write Semantics
4. Replica set in practice

MongoDB write path



MongoDB Journal vs Oplog

- **journal**
  - low level log of an operation for crash recovery (can be turned off)
- **oplog**
  - similar to RDBMS binlog
    - stores (idempotent) high-level transactions that modify the database
    - kept on the master and used for replication

https://docs.mongodb.org/manual/core/read-isolation-consistency-recency/

MongoDB Replication

1. MongoDB write path
2. **Replication principles**
3. Replica set Read and Write Semantics
4. Replica set in practice

Replica set

- **Replica set** = a group of *mongod processes* that provide **redundancy** and **high availability**
- Writes: write to single node replicated to the others members of the replica set
- Read: read from a single member of the replica set

Disclaimer:

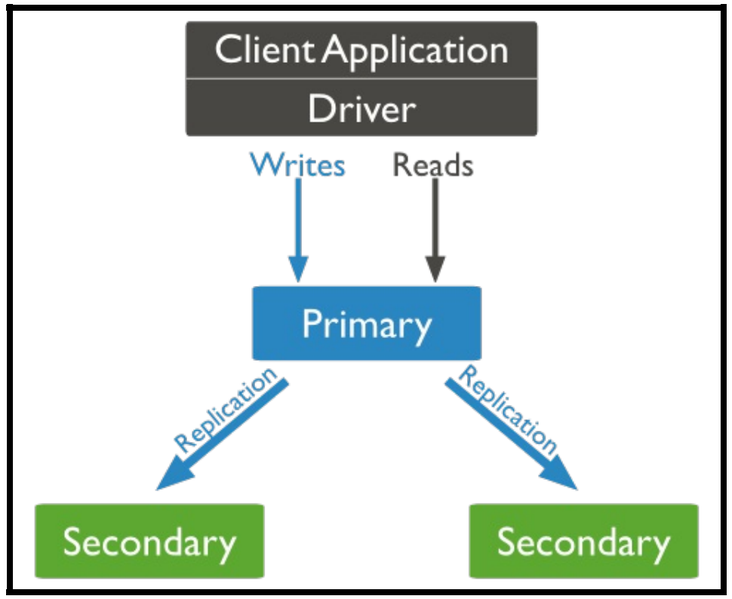
- we only consider replica sets **without sharding** (for now)
- we not include proposed MongoDB 3.2 replication modifications (readConcern…)

Replica set members

- **Primary**
  - acceptes all **writes** and reads
  - 1 primary per replica set
- **Secondaries** replicates data (and can serve **reads** ⇒ reads preferences)
  - Priority 0 ⇒ Hidden members ⇒ Delayed
- **Arbiters** (usually at most one) : break ties

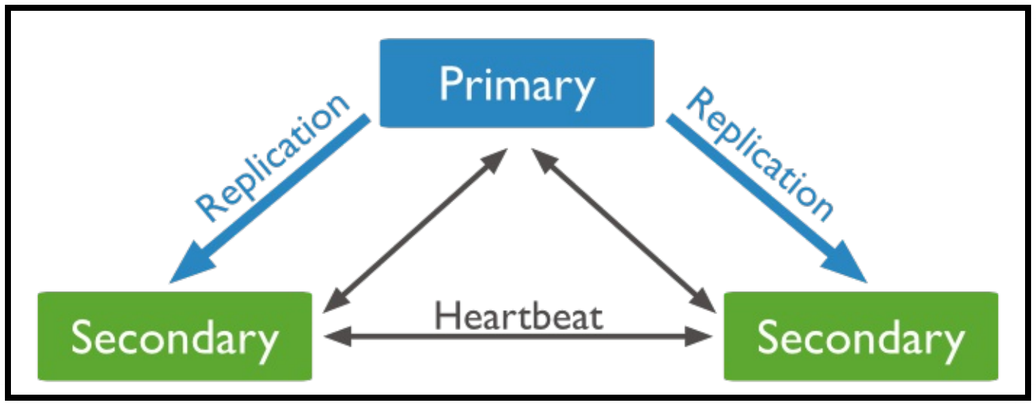
Primary and secondary members

- **Primary** acceptes all **writes** + reads + records them in oplog
- **Secondary** replicates primary oplogs (also accept reads)

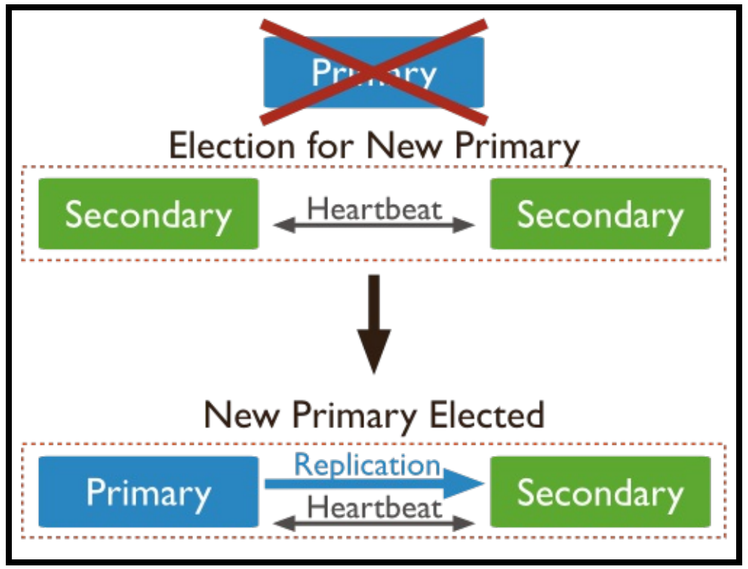


Replication data flow

- asynchronous **oplog** replication
- heartbeat for monitoring status



Automatic failover via new primary election

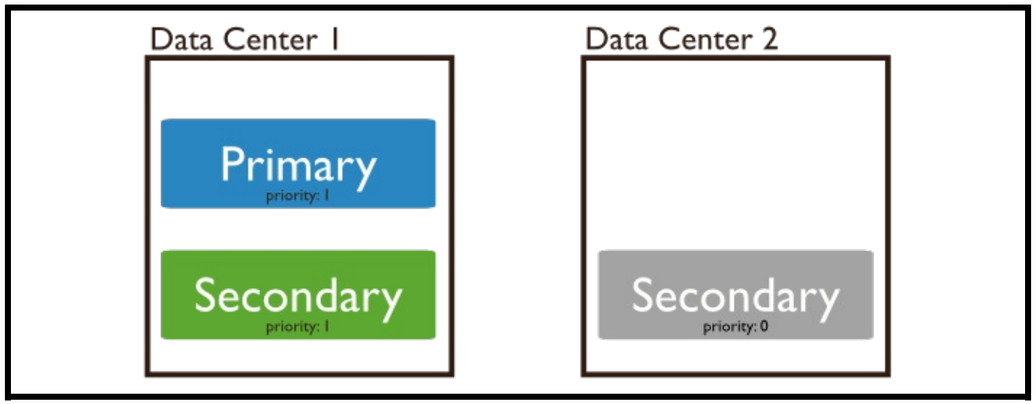


Strategy for election

- **member’s priority**
- **latest optime in the oplog**
- uptime
- break the tie rules

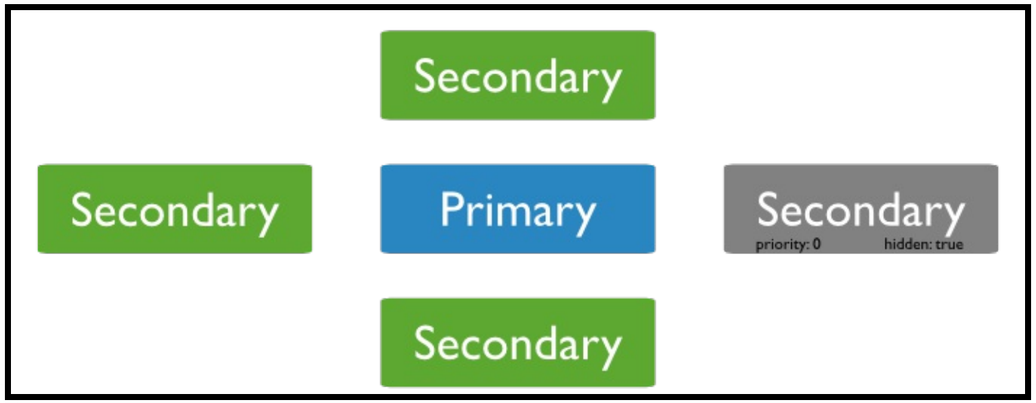
Secondary members: Priority 0

- cannot become primary
- cannot trigger elections
- can vote in elections
- copy of data + accepts reads



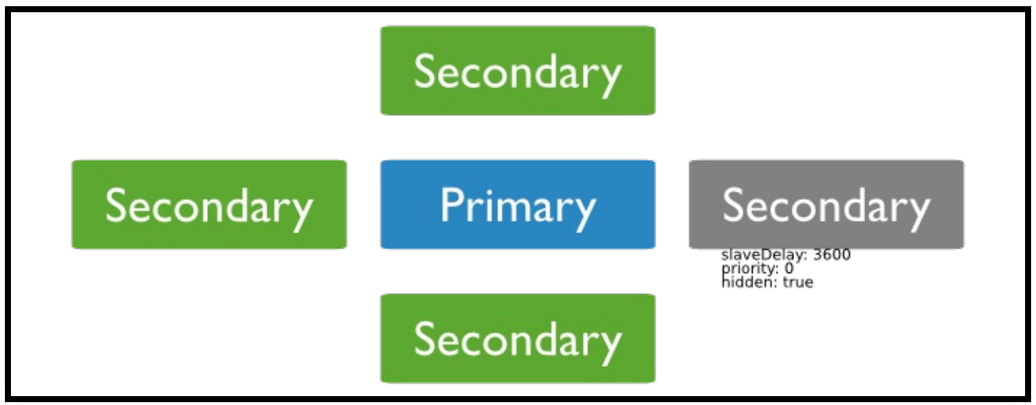
Secondary members: Hidden replica set member

- Priority 0 members that don't accept reads



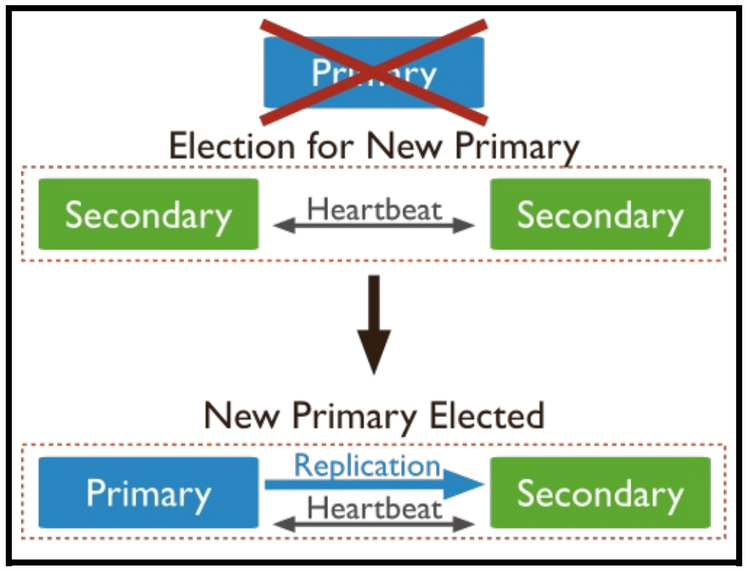
Secondary members: Delayed replica set members

- reflect an delayed state of the set
  - **must be priority 0** ⇒ prevent them to become primary
  - **should be hidden** ⇒ prevent application to query stale data



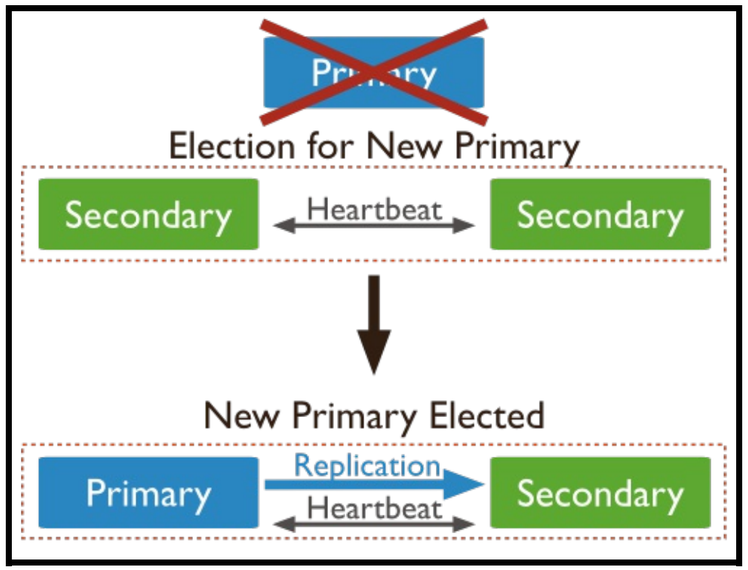
Elections on odd number of nodes

- a replica cannot become primary with only 1 vote
- majority with even numbers of members ?



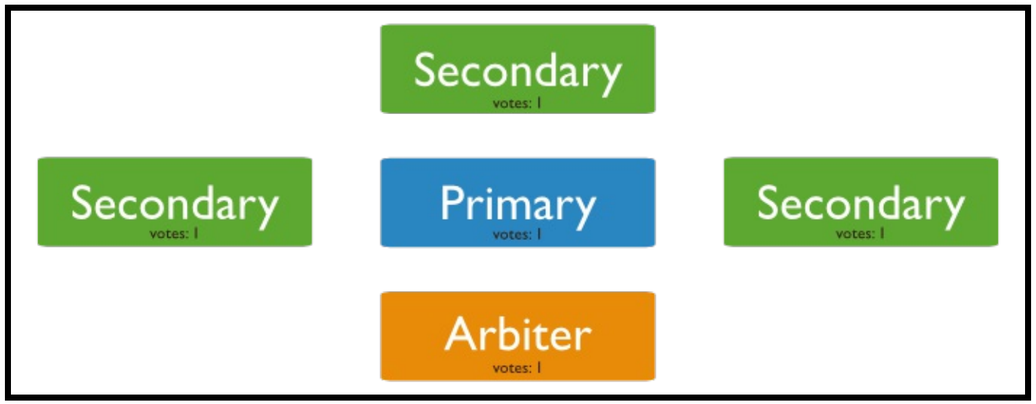
Elections on odd number of nodes

- a replica cannot become primary with only 1 vote
- majority with even numbers of members ?



- use **Arbitrers** to break ties
  - does not hold data
  - cannot became a primary

Arbiters


Secondary
votes: 1
Secondary
votes: 1
Primary
votes: 1
Secondary
votes: 1
Arbiter
votes: 1

Fault tolerance

- **No primary** ⇒ writes no longer possible, reads still accepted
- **Fault tolerance** : number of members that can become unavailable and still be able to elect a primary

Number of members	Majority required to elect a primary	Fault tolerance
3	2	1
4	3	1
5	3	2
6	4	2

```
https://docs.mongodb.org/manual/core/replica-set-architectures/
```

Rollbacks during replica set failover

- a rollback reverts write operations on a former primary when the member rejoins its replica set after a failover
  - the primary accepted a write that was not sucessfuly replicated to secondaries !

Cause of the problem ?

default write semantics { w:1 } ⇒ the primary acknowledge the write after the local write (local Journal!)

How to handle rollbacks

- manually apply/discard rollbacks (**rollback/** folder)
- *avoid* rollbacks use **{ w:majority }**
  - READ UNCOMMITED SEMANTICS
    - ! Regardless of write concern, other clients can see the result of the write operations before the write operation is acknowledged to the issuing client.
    - ! Clients can read data which may be subsequently rolled back.

https://docs.mongodb.org/manual/core/replica-set-rollbacks/

https://docs.mongodb.org/manual/core/read-isolation-consistency-recency/

MongoDB Replication

1. MongoDB write path
2. Replication principles
3. **Replica set Read and Write Semantics**
    1. Write concerns
    2. Read preferences
4. Replica set in practice

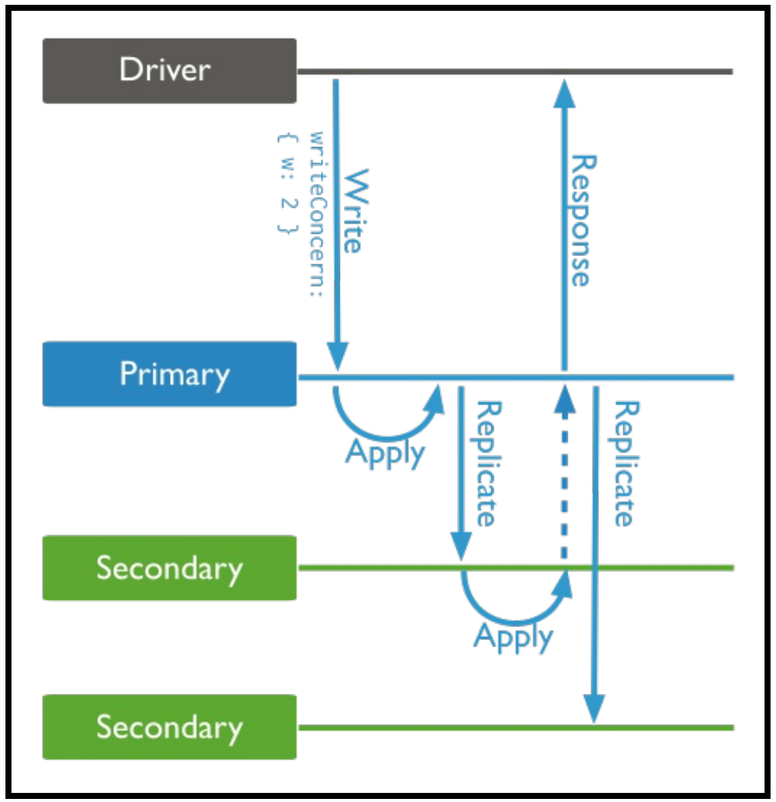
Replica set Read and Write Semantics

- parameters that change the default read/write semantics (**move** the *CAP cursor*)
  - **write concern**
    - is the guarantee an application requires from MongoDB to consider a write operation successful
  - **read preference**
    - applications specify read preference to control how drivers **direct read operations** to members of the replica set

Write semantics

- **w:1** (*default*)
  - the primary acknowledge the write after the local write
- other options:
  - **w:N**
    - ack the write after the ack of N members
  - **w:majority**
    - ack the write after the ack of the majority of the members
- optional parameter **wtimeout**
  - prevents write operations from blocking indefinitely if the write concern is unachievable

W:2 write semantics



Changing the write semantics

- at the query level

```
db.products.insert(
   { item: "envelopes", qty : 100, type: "Clasp" },
   { writeConcern: { w: 2, wtimeout: 5000 } }
)
```

- change the default write concern:

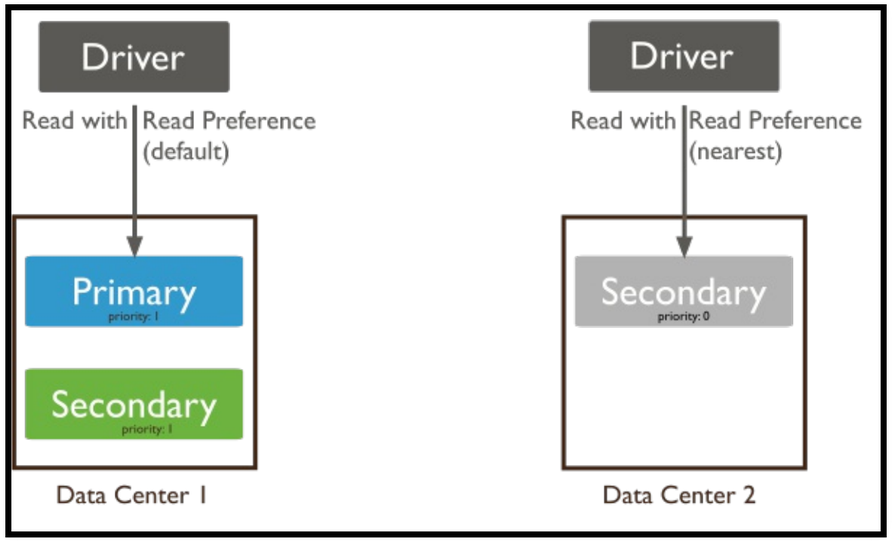
```
cfg = rs.conf()
cfg.settings = {}
cfg.settings.getLastErrorDefaults = { w: "majority", wtimeout: 5000 }
rs.reconfig(cfg)
```

Read preference

- **primary** (*default*)
  - read from the current replica set primary.
- **primaryPreferred**
  - read from primary (or secondary iff no primary)
- **secondary**
  - read from secondary members
- **secondaryPreferred**
  - read from secondary(or primary iff no secondary)
- **nearest**
  - read from the member with the least network latency

Async replication ⇒ stale data if read from replica

Read preferences example



Read preferences use cases

- **Maximize Consistency** ⇒ **primary** read preference
- **Maximize Availability** ⇒ **primaryPreferred** read preference
- **Minimize Latency** ⇒ **nearest** read preference

MongoDB Replication

1. MongoDB write path
2. Replication principles
3. Replica set Read and Write Semantics
4. **Replica set in practice**

MongoDB consistency in real world

> *Read the documentation for the systems you depend on thoroughly–then verify their claims for yourself. You may discover surprising results!*

— Kyle Kingsbury(Aphyr)

https://aphyr.com/posts/322-jepsen-mongodb-stale-reads

Learn more:

- read the MongoDB documentation and the Jespen blog entry:
  - MongoDB Documentation
  - Jepsen MongoDB Stale reads on
- do the replica set tutorial in the MongoDB documentation:
  - https://docs.mongodb.org/manual/administration/replica-set-deployment/