MongoDB Replication

COSC 061 - Winter 2025 - Dartmouth College

MongoDB Replication

- Storing multiple copies of data across several servers and keeping them synchronized
- Fault tolerance against the loss of a server
- Each node is a "replica set member"
- Fault tolerance, high availability, and data durability

Components

- 3, 5, 7, or up to 50 `mongod` servers
- Each server is a "node"
- One "Primary" and the rest are "Secondaries"

Components: The Primary

One Primary: the only node that can receive write operations

- Writes are recorded in the Primary and noted in its operations log (**oplog**)
- By default, it also handles all Read operations
- There is an option to allow secondaries to also handle Reads

Components: The Secondaries

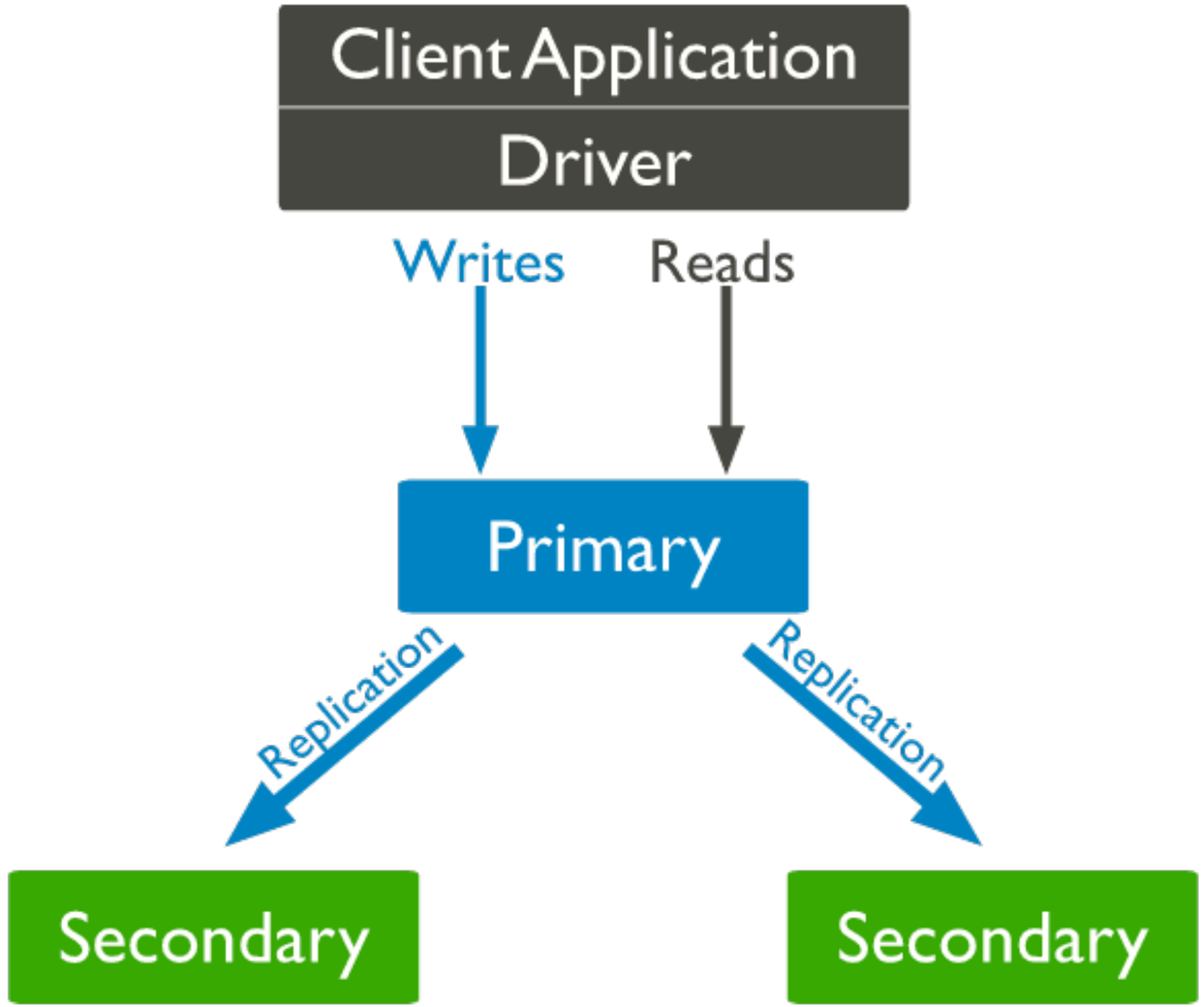
A replica set member that replicates the contents of the primary member

It duplicates the Primary's oplog entries and applies the operations to their own data set.

Replica Sets

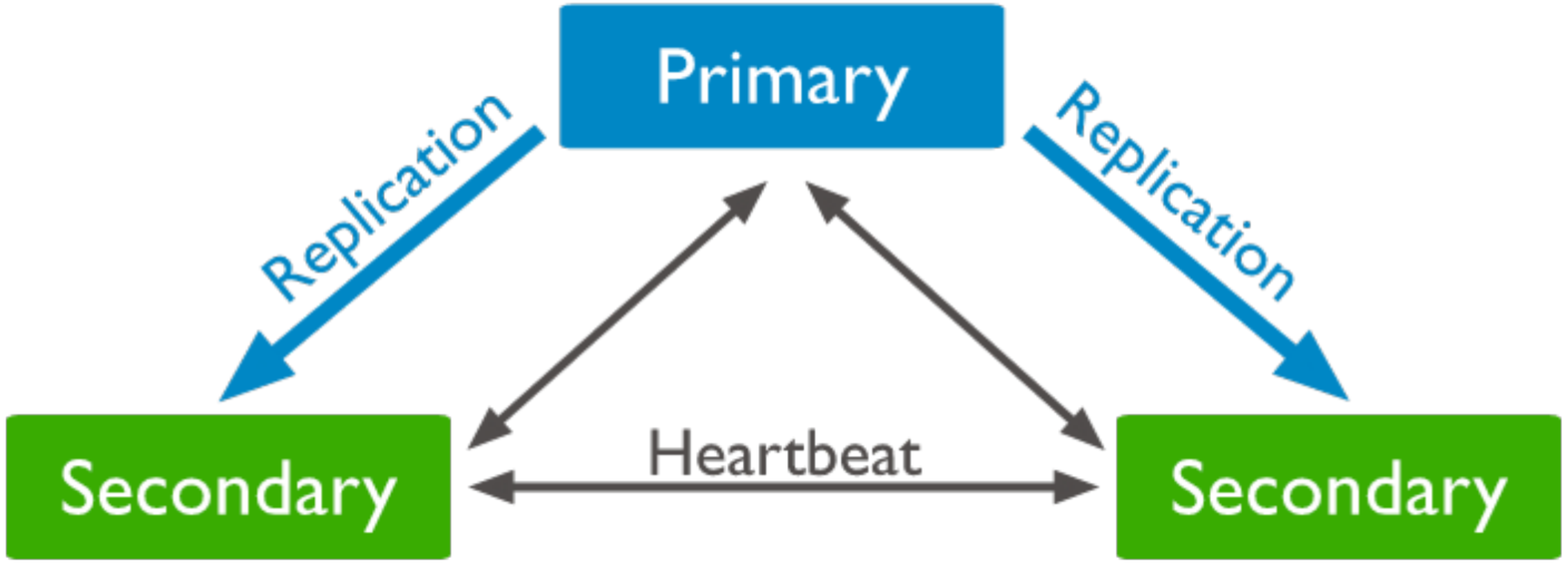
What's behind the curtain

- Your app uses the driver to interact with the Replica set.
- The primary receives all write operations.
- Secondaries replicate operations from the primary to maintain an identical data set.



Replica Sets

Ready for problems

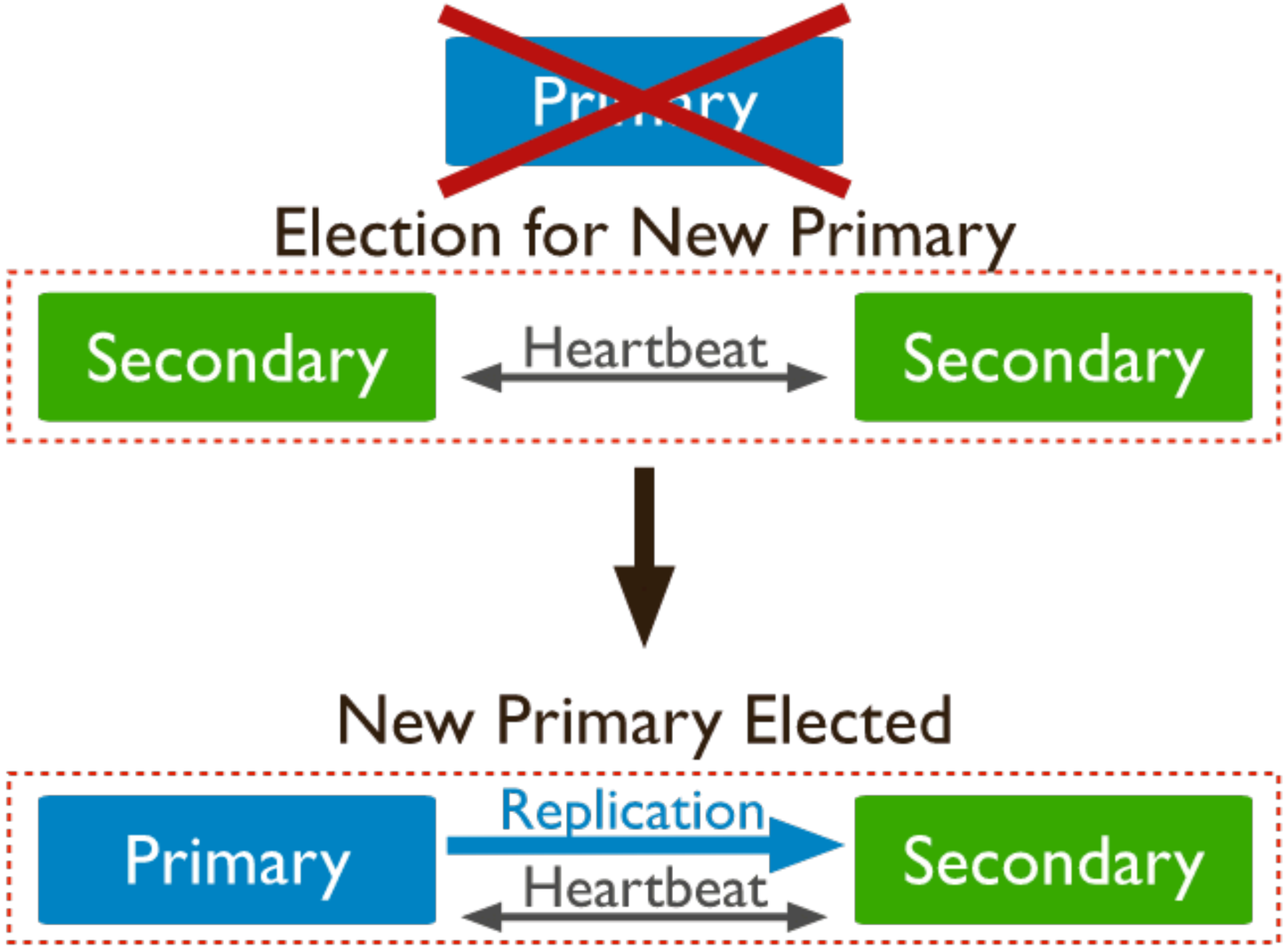


- Writes are applied to the Primary and added to its oplog
- Secondaries replicate the oplog and apply writes to their data sets
- All can accept read requests, but by default reads go to the Primary

Replica Sets

Failover operation

- If the primary becomes unavailable, a secondary calls for an election.
- Any Secondary can become the new Primary, subject to some design constraints.



What can initiate an election ?

- When the primary becomes unavailable
- When a new node is added to the replica set
- When a replica is purposely taken offline for maintenance
  - e.g., via `rs.stepDown( )` or `rs.reconfig( )` commands
- When Secondaries lose connectivity with the Primary for more than some configured amount of time (default is 10s.)

Voting rules

- Each voting member casts one vote per election
- A maximum of seven members can have voting privileges
- It's important to have an odd number of voting members in the replicas set to ensure a decision
- It's recommended that a rep set have 3, 5, or 7 members

Election start

The secondary that called for the election starts it by:

- Sharing how recent its data is, and
- Sharing the current election term (a count of the number of election so far)

It then proceeds to vote for itself

Replica set member priority

- Each member is assigned a priority that represents its priority for becoming the Primary during an election
- Default priority is 1
- Range of priorities from 0 to 1000 can be assigned to reflect server size, capacity, MTBF, location, or similar attributes
  - Priority 0 can never initiate an election or be voted in as the Primary

If the 1st Primary rejoins

- 1st Primary may later rejoin the replica set
- It catches up to the others by applying all the operations that it missed
- Once up to date, it becomes an equal partner with the other Secondaries.

The oplog

- A special collection that's defined as a "capped" collection
- That means it operates like a "Ring Buffer"
  - The oldest entries in this special collection are overwritten once it reaches capacity.
- It enables a member to recover the updates since a specified timestamp
- It can be used to identify Secondaries lagging behind the Primary

oplog updates

- With every update to the database, the Primary performs the update and notes the changes in the oplog
- The Secondaries are continuously pulling oplog updates from the Primary
  - Applying those changes to their copy of the database
  - Updating their own oplog to reflect what was done
- Entries are “idempotent”, meaning they can be applied any number of times with the same final result

Replica set info

```
rs.printReplicationInfo( )
```

shows info about oplog

```
rs.printSecondaryReplicationInfo( )
```

shows info about all the secondaries, including "rep lag"

- replication lag (rep lag) is how far behind a secondary is w.r.t. the primary
- If a member gets too far behind, it goes into “Recovery”
  - it can still vote but cannot accept Read requests
  - it then begins an initial sync to essentially restart: copying all data and the oplog from a real set member … expensive

Replica Sets

Read Preference

- If an app prefers to read from a secondary, that's ok.