UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

**Replikacija podataka kod MongoDB baza podataka**

Seminarski rad iz predmeta

Sistemi za upravljanje bazama podataka

Student: Mentor:

Marta Đorđević, 1490 prof. dr Aleksandar Stanimirović

Niš, 2023. godina

Sadržaj

[Uvod 3](#_Toc138800029)

[Mehanizam replikacije 4](#_Toc138800030)

[Skup replika 4](#_Toc138800031)

[Primarni čvor 4](#_Toc138800032)

[Sekundarni čvorovi 5](#_Toc138800033)

[Čvor sa prioritetom 0 6](#_Toc138800034)

[Primer tročlanog seta replika 7](#_Toc138800035)

[Primer petočlanog skupa replika 7](#_Toc138800036)

[Izbor članova 7](#_Toc138800037)

[Skriveni čvorovi 8](#_Toc138800038)

[Odloženi čvorovi 9](#_Toc138800039)

[Arbitražni čvor 9](#_Toc138800040)

[Oplog 10](#_Toc138800041)

[Sinhronizacija 10](#_Toc138800042)

[Inicijalna sinhronizacija 10](#_Toc138800043)

[Replikacija 11](#_Toc138800044)

[Write concern 12](#_Toc138800045)

[Read preference 13](#_Toc138800046)

[Zaključak 15](#_Toc138800047)

# Uvod

U današnje vreme pouzdanost i dostupnost podataka su ključni faktor za uspeh mnogih organizacija. Bilo da se radi o velikim kompanijama, pa do malih preduzeća, gubitak i nedostupnost podataka može prouzrokovati ogromne posledice. Kako ne bi došlo do ovakvih situacija, baze podataka nude različite tehnike replikacije podataka, a MongoDB je jedan od vodećih koji pruža skalabilne mehanizme za replikaciju.

Replikacija podataka u MongoDB-u omogućava stvaranje kopija podataka i njihovo raspoređivanje na više čvorova ili servera unutar baze podataka. Na ovaj način se održava pouzdanost i dostupnost podataka čak i u slučaju kvara ili iznenadnog prekida rada jednog od čvorova.

Kroz ovaj rad biće prikazane prednosti replikacije podataka u MongoDB-u i to kako se pomoću ove tehnike održava pouzdanost, skalabilnost i dostupnost podataka. Replikacija postaje ključni alat za svaku organizaciju koja se oslanja na podatke kakko bi održala konkurentnost i zadovoljila visoke zahteve korisnika.

# Mehanizam replikacije

MongoDB mehanizam replikacije pruža sinhronizaciju podataka između većeg broja MongoDB instanci. Replikacija omogućava da se poveća dostupnost podataka na taj način što se kreiraju više kopija podataka na serverima. Ova funkcionalnost omogućava veću dostupnost podataka, povećava otpornost na kvarove i povećava skalabilnost sistema. Replikacija obezbeđuje redudantnost i povećava dostupnost podataka kod čitanja. Trajnost podataka se održava tako što se čuva više kopija ili replika tih podataka na fizički izolovanim serverima.

## Skup replika

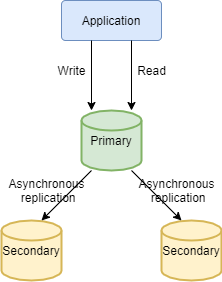
Skup replika je grupa mongod procesa koji održavaju isti skup podataka. Jedna instanca je primarna, a sve ostale su sekundarne. Svi klijenti upisuju podatke u primarnu instancu, a podaci se nakon toga asinhrono sinhronizuju sa sekundarnim instancama. Minimalni broj čvorova u jednom skupu replika je 3, jedan primarni i dva sekundarna. Skup replika može imati najviše 50 članova, ali samo 7 članova tog skupa ima pravo glasa.



Dodavanje člana skupu replika

### Primarni čvor

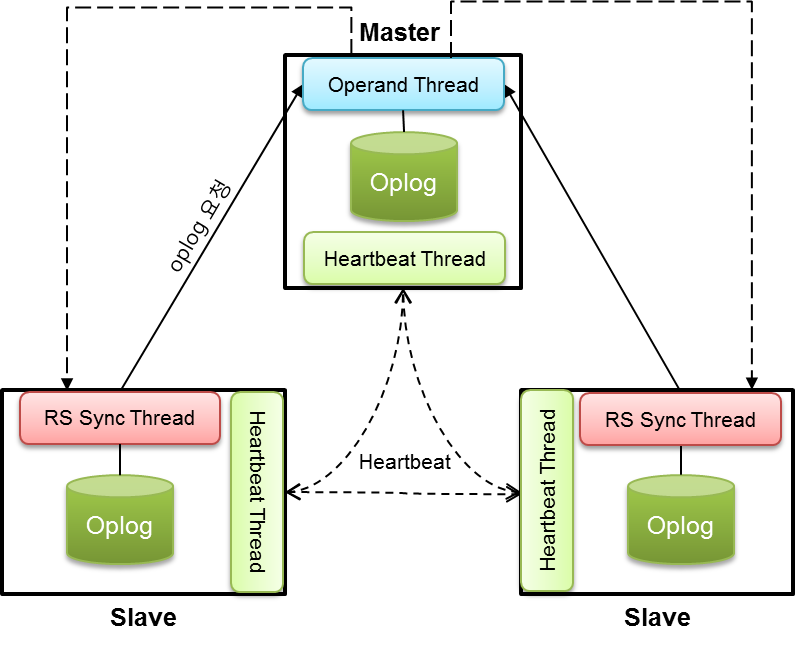
Primarni čvor je jedini čvor u skupu replika koji može da prima operacije pisanja, dok je operacija čitanja moguća na svim čvorovima iz skupa replika. MongoDB primenjuje operacije pisanja na primarnom čvoru, odnosno beleži operacije na oplogu primarnog čvora, a nakon toga se ovaj dnevnik kopira na sekundarnim članovima i primenjuju se operacije iz oploga na svoje skupove podataka. Skup replika može imati najviše jedan primarni čvor. U slučaju da primarni čvor postane nedostupan, pokreće se glasanje gde sekundarni čvorovi biraju novi primarni čvor.



Slika 1 – Primarni čvor

### Sekundarni čvorovi

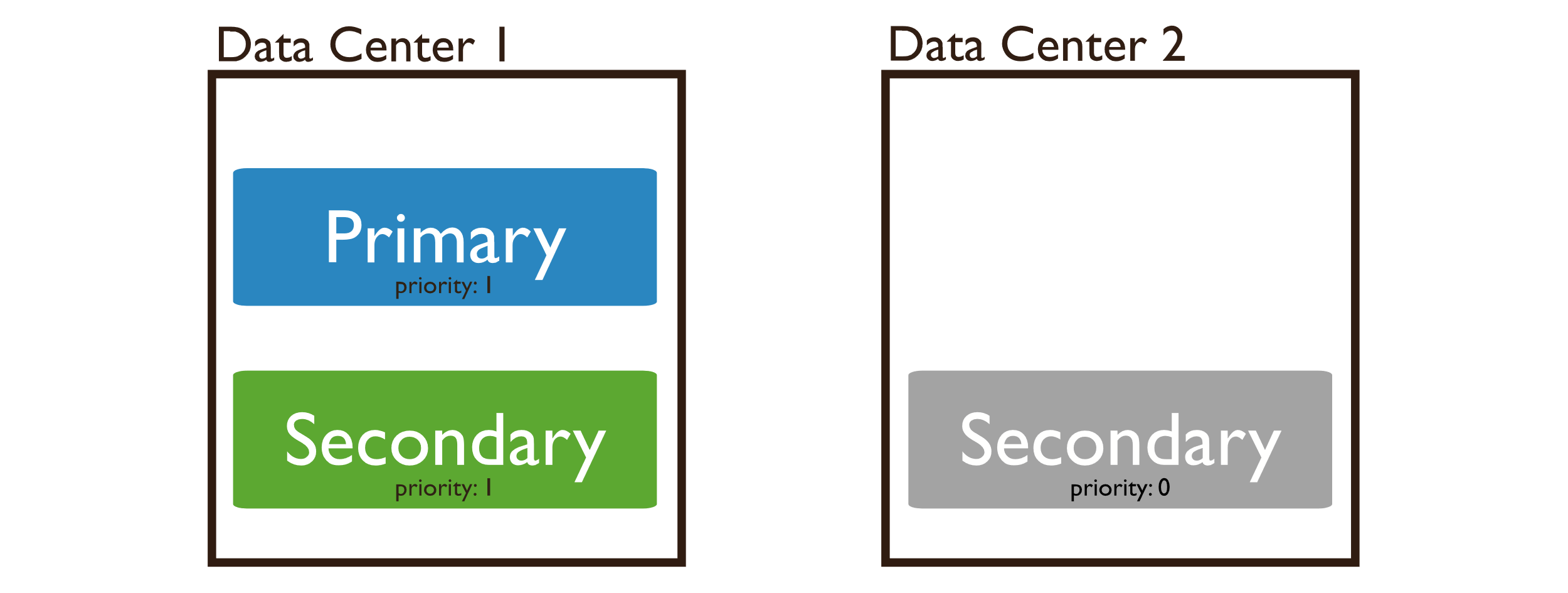
Sekundarni čvorovi održavaju kopiju skupa podataka primarnog čvora. Da bi kopirao podatke sa primarnog, sekundarni čvor primenjuje operacije iz oploga primarnog čvora na sopstveni skup podataka u asinhronom procesu. Skup replika može imati jedan ili više sekundarnih čvorova. Ako trenutni primarni čvor otkaže, metodom izbora bira se sekundarni čvor koji će preuzeti ulogu primarnog čvora i na taj način se sprečava gubljenje podataka usled kvarova i otkaza. **Proces otkucaja srca** je proces kojim se identifikuje trenutni status čvora u skupu replika. Čvorovi u skupu replika šalju pingove međusobno svake dve sekunde, a u slučaju da neki čvor ne uzvrati pingom u roku od 10 sekundi biće označen za nedostupan. Ovo je od velikog značaja za proces automatskog pronalaska greške i nakon što se uoči greška neki od sekundarnih čvorova će postati primarni.



Slika 2 – Skup replika sa procesom otkucaja srca

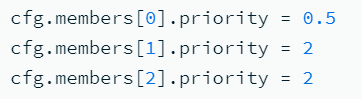
### Čvor sa prioritetom 0

Čvor sa prioritetom 0 je čvor koji ne može postati primarni čvor i ne može pokrenuti izbore. Taj čvor neće biti kandidat u izboru za primarni čvor, ne može postati primarni i ne može pokrenuti glasanje. Izuzev ovoga, oni funkcionišu kao normalni sekundarni čvorovi koji imaju kopije podataka, podržavaju operacije čitanja i glasaju na izborima. Čvorovi sa prioritetom 0 se uvode kada postoje članovi koji se nalaze u centru podataka koji je udaljen od glavne primene i zbog toga ima veće kašnjenje. Ovakvi čvorovi odgovaraju lokalnim zahtevima za čitanje, ali nisu dobri da obavljaju ulogu primarnog čvora. Na slici 3 je dijagram koji prikazuje centar podataka sa leve strane koji hostuje primarni i sekundarni čvor i centar podataka sa desne strane koji hostuje sekundarni čvor sa prioritetom 0. Zbog ovakve postavke, samo članovi levog centra podataka mogu postati primarni čvor.



Slika 3 – Dva centra podataka

Dok skupovi replika obezbeđuju zaštitu od gubljenja podataka prilikom otkaza primarnog čvora, skupovi replika koji se nalaze u jednom centru podataka podložni su greškama centra podataka. Nestajanje struje, prekidi u mreži i prirodne katastrofe mogu dovesti do problema. Distribucija članova skupa replika u geografski odvojene centre podataka smanjuje posledice grešaka ako jedan od centara podataka postane nedostupan. Da bi se zaštitili podaci u slučaju kvara centra podatakam, bar jedan od članova treba ostati u alternativnom centru podataka i bilo bi dobro da broj centara podataka bude neparan broj.



Dodeljivanje prioriteta



Onemogućavanje sekundarnog da postane primarni

## Primer tročlanog seta replika

Ako postoji tročlani skup repelika, članove je moguće distribuirati na sledeće načine:

* Dva centra podataka – dva člana bi se nalazila u centru podataka 1, a jedan član u centru podataka 2. Ukoliko je jedan od članova arbitražni čvor bilo bi dobro da se on nalazi u centru podataka 1 zajedno sa primarnim čvorom koji nosi podatke. Ako centar podataka 1 prestane sa radom, skup replika počinje da služi samo za operacije čitanja, jer su operacije pisanja moguće samo na primarnom čvoru. Ako prestane sa radom centar podataka 2, skup replika i dalje može da prima operacije čitanja i pisanja.
* Tri centra podataka – sva tri člana skupa replika idu u poseban centar podataka. Ako bilo koji centar podaka postane nedostupan, skup replika i dalje zadržava mogućnost pisanja jer preostali članovi mogu održati izbore.

Dobra praksa je da se članovi skupa replika distribuiraju u bar dva centra podataka, jer u slučaju da se jedan pokvari podaci su i dalje dostupni za čitanje. Ako postane nedostupan onaj centar podataka koji ima manje članova, skup replika i dalje može služiti za pisanje i za čitanje, međutim ako postane nedostupan onaj centar sa više članova pisanje postaje nemoguće.

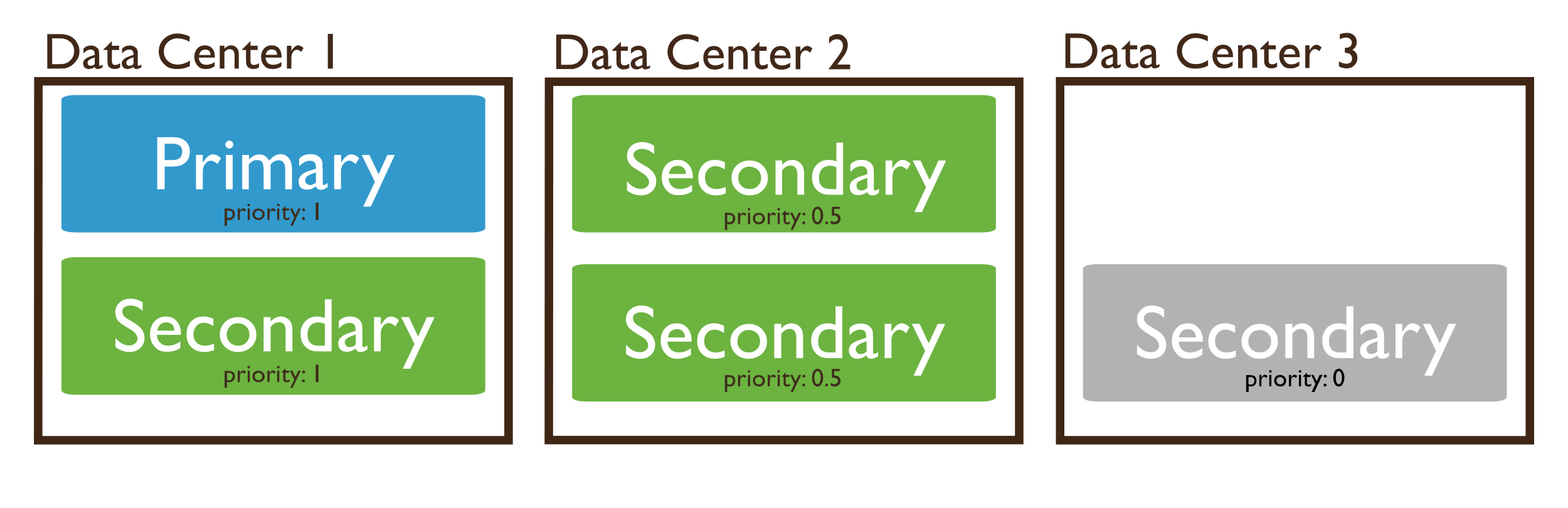
## Primer petočlanog skupa replika

Mogući načini distribucije:

* Dva centra podataka – tri člana idu u centar podataka 1, a dva člana u centar podataka 2. Ako centar podataka 1 prestane sa radom, skup replika će primati samo operacije čitanja, jer se u njemu nalazi većina, a ako centar podataka 2 prestane, skup replika će primati i pisanje.
* Tri centra podataka – po dva člana u centre podataka 1 i 2 i jedan član u centar podataka 3. Ako se neki od centara pokvari, skup replika ostaje i za operacije pisanja.

## Izbor članova

Neki članovi skupa replika, kao što su članovi koji imaju ograničenje mreže ili ograničene resure, ne bi trebalo da postanu primarni čvorovi nikada i zato njima treba staviti prioritet 0. Ako bi bilo bolje da članovi jednog centra budu izabrani za primarne od nekog drugog centra, onda njima treba postaviti veći prioritet. Na slici je prikazano kako članovi centra podataka 1 imaju najveći prioritet, a član centra 3 ne može postati primarni čvor.

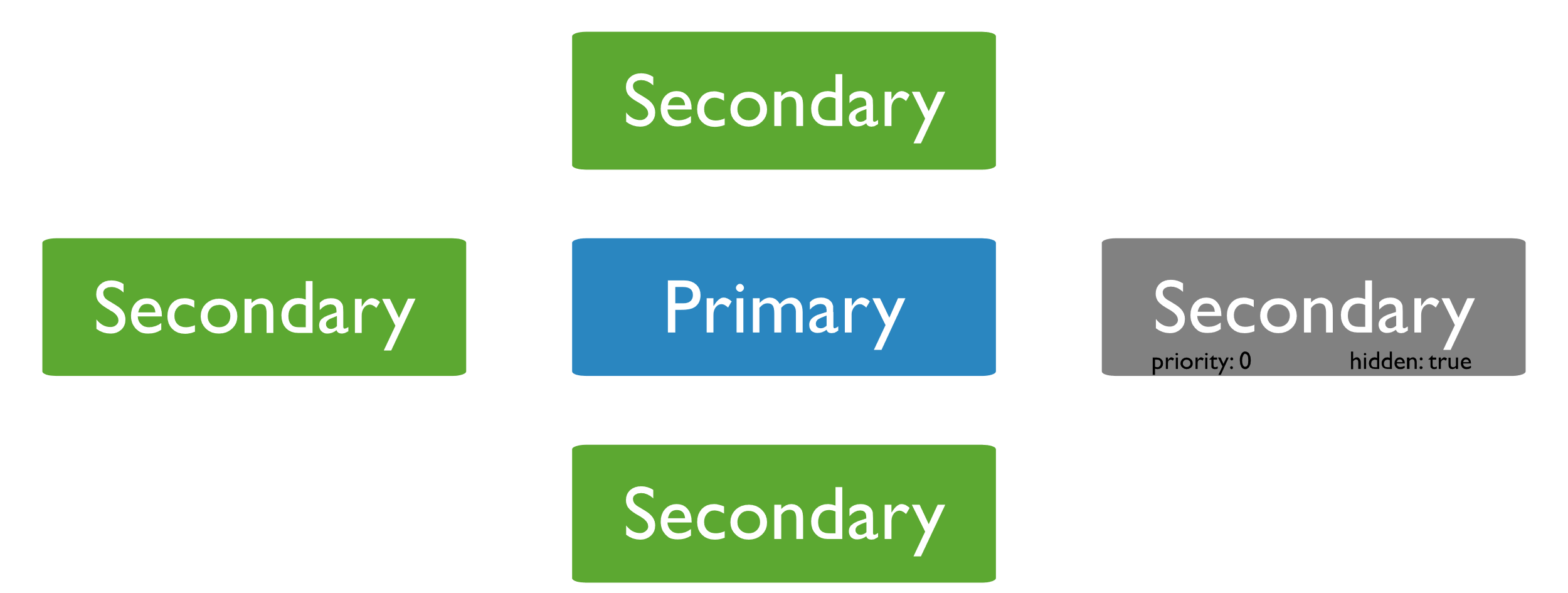


Slika 4 – Različiti prioriteti

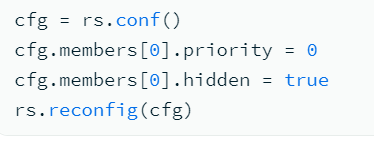
### Skriveni čvorovi

Poseban tip sekundarnih čvorova koji održavaju kopiju primarnog čvora, ali su konfigurisani tako da nisu vidljivi klijentskim aplikacijama za izvršenje operacija čitanja. Oni uvek moraju imati prioritet 0 na taj način ne mogu postati primarni čvorovi. Time što su nevidljivi za klijentske zahteve smanjuje se opterećenje replikacijskog skupa i poboljšavaju performanse. Iako su skrivene, one i dalje učestvuju u replikaciji i glasanju i često se koriste za obavljanje analitičkih operacija koje ne utiču na performanse primarnog čvora i klijentskih aplikacija. Metoda *db.hello()* ne prikazuje skrivene članove.

Na slici 5 prikazan je skup replika od 5 članova, gde 4 sekundarne replike imaju kopije primarnog čvora, ali je jedna replika skrivena. Kao rezultat toga, ovaj član neće dobiti nikakav saobraćaj osim osnovne replikacije. Skrivene članove je dobro koristiti za namenske operacije, kao što su pravljenje rezervnih kopija i izveštavanje.

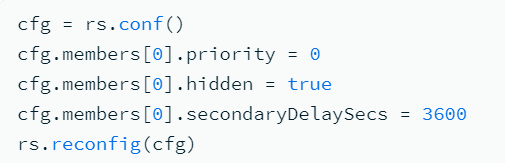


Slika 5 – Skrivene replike



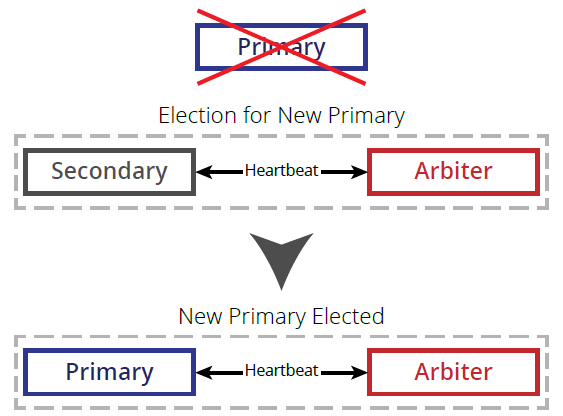
### Odloženi čvorovi

To je tip sekundarnih replika koje su konfigurisane tako da namerno kasne u primanju operacija pisanja od strane primarnog čvora i replikaciji u cilju oporavljanja od različitih grešeka. Da odloženi član ne bi postao primarni, potrebno mu je staviti prioritet 0 i da budu skrivene. Sakrivanje odloženih članova skupa replika sprečava aplikacije da traže podatke. Kada se čvor konfiguriše kao odloženi, postavlja se vremensko kašnjenje između trenutka kada primarni čvor prima pisane operacije i trenutka kada se te operacije kopiraju na odloženu sekundarnu repliku, pa odložena replika neće odmah imati promene sa primarnog čvora. Kada se bira kašnjenje, treba uzeti u obzir da iznos kašnjenja mora biti jednak ili veći od očekivanog perioda održavanja.



### Arbitražni čvor

U nekim situacijama kada postoje primarni i sekundarni čvorovi, ali je zbog troškova ograničeno dodavanje drugog sekundarnog, može postojati arbitražni čvor u skupu replika. Ovaj čvor učestvuje na izborima za primarne čvorove, ali on nema kopiju skupa podataka i nikada ne može postati primarni. Kako bi se izbegli problemi sa konzistentnošću podataka, trebalo bi da postoji samo jedan arbitražni čvor. Da bi se osiguralo da će upis trajati nakon kvara primarnog čvora, većina čvorova treba da potvrdi operaciju pisanja. Arbitri ne čuvaju nikakve podatke, ali doprinose broju čvorova u skupu replika. Ako skup replika ima više arbitražnih čvorova, manja je verovatnoća da će većina čvora biti dostupna nakon kvara. Pomoću njega se štede resursi jer ne zahteva dodatni prostor na disku potreban za replikaciju podataka jer ne sadrži kopije podataka, te postojanje arbitarskog čvora ne povećava opterećenje sistema. Uloga ovog čvora ne povećava otpornost na kvarove niti povećava dostupnost podataka jer se ne radi o kopiji, već samo pomaže u izboru novog primarnog čvora. Jedina komunikacija između arbitražnog čvora i ostalih članova skupa su glasovi tokom izbora, proces otkucaja srca i podaci o konfiguraciji.



Slika 6 – Postojanje arbitražnog čvora



## Oplog

Oplog (dnevnik operacija) je posebna ograničena kolekcija koja vodi evidenciju o operacijama koje menjaju podatke uskladištene u bazama podataka. MongoDB primenjuje operacije baze podataka na primarnom čvoru, a zatim te operacije zapisuje na oplogu primarnog. Nakon toga ove operacije se kopiraju i preminjuju na sekundarnim čvorovima u asinhronom procesu. Svi članovi skupa replika sadrže kopiju oploga i to im omogućava da održavaju trenutno stanje baze.

## Sinhronizacija

Da bi se održale ažurne kopije deljenog skupa podataka, sekundarni članovi skupa repelika se sinhronizuju. U MongoDB-u postoje dva tipa sinhronizacije podataka: inicijalna sinhronizacija i replikacija tekućih promena na ceo skup podataka.

### Inicijalna sinhronizacija

Inicijalna sinhronizacija kopira sve podatke sa jednog člana skupa replika na drugi član. Od verzije MongoDB 4.4 moguće je navesti početni izvor sinhronizacije koristeći parametar *initialSyncSourceReadPreference* i ovaj parametar moguće je specificirati samo kada se pokreće *mongod*. Od verzije MongoDB 5.2, inicijalne sinhronizacije mogu biti logčke ili zasnovane na kopiranju datoteka.

#### Logička inicijalna sinhronizacija

Kloniraju se sve baze podataka osim lokalne baze podataka. Da bi se klonirale, mongod skenira svaku kolekciju u svakoj izvornoj bazi podataka i upisuje sve te podatke u sopstvene kopije ovih kolekcija. Takođe, za svaku kolekciju pravi indeksne kolekcije i povlači novododate oplog zapise tokom kopiranja. Bitno je da član u koji se kopira ima dovoljno prostora na disku u lokalnoj bazi podataka kako bi privremeno uskladištio oploge tokom kopiranja. Koristeći oplog onog čvora iz koga se kopira, mongod ažurira svoj skup podataka kako bi se održavalo trenutno stanje.

#### Inicijalna sinronizacija zasnovana na kopiranju datoteka

Inicijalna sinronizacija zasnovana na kopiranju datoteka pokreće početni proces sinhronizacije kopiranjem i premeštanjem datoteka u sistem datoteka. Ovaj metod sinronizacije može biti brži od logičke inicijalne sinhronizacije. Da bi se omogućio ovaj tip sinhronizacije, potrebno je parametar *initialSyncMethod* postaviti na *fileCopyBased* na odredišnom članu. Ovaj parametar se može podesiti samo prilikom pokretanja. Inicijalna sinhronizacija zasnovana na kopiranju datoteka zamenjuje lokalnu bazu podataka sa lokalnom bazom podataka člana sa koga se sinhronizuje. Ograničenje je da tokom ovog procesa nije moguće pokrenuti rezervnu kopiju člana sa kojeg se sinhronizuje.

#### Tolerancija na greške

Ako sekundarna replika koja vrši sihronizaciju naiđe na grešku tokom procesa sinhronizacije, on ponovo pokreće sinhronizaciju od početka. Od verzije MongoDB 4.4, sekundarna inicijalna sinhronizacija može pokušati da nastavi proces sinhronizacije ako je prekinuta greškom ili pod nekim drugim uslovom. Sekundarna replika pokušava da nastavi inicijalnu sinhronizaciju tokom 24h. Dodaje se parametar *initialSyncTransientErrorRetryPeriodSeconds* za kontrolu količine vremena tokom kojem sekundarna replika pokušava da nastavi inicijalnu sinhronizaciju. U slučaju da nije moguće da uspešno nastavi proces sinhronizacije tokom ovog vremena, bira se novi izvor iz skupa replika i ponovo pokreće proces sinhronizacije.

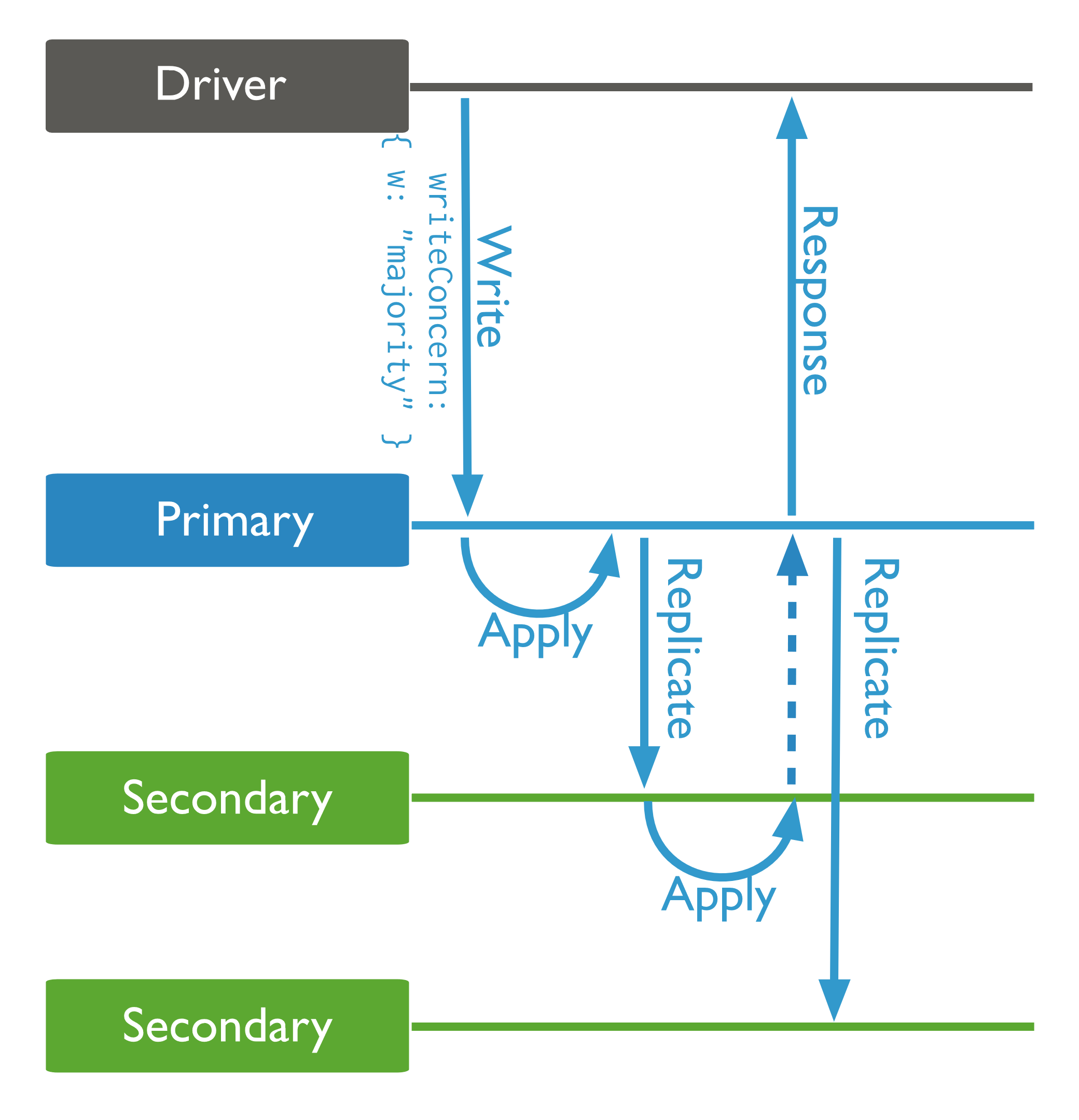
### Replikacija

Sekundarni članovi kontinuirano kopiraju podatke nakon inicijalne sinhronizacije. Oni kopiraju oplog primarnog i te operacije primenjuju u asinhronom procesu. MongoDB primenjuje operacije pisanja u serijama koristeći više niti. Pravi grupe prema ID-u dokumenta i istovremeno primenjuje grupe operacija koristeći drugu nit. Operacije pisanja se uvek primenjuju u onom redosledu kako su u originalu primenjene. Čitanje iz snimaka obezbeđuje konzistentan prikaz podataka i omogućava da se čitanje odvija istovremeno sa tekućom replikacijom bez potrebe za zaključavanjem. Sa verzijom MongoDB 4.2, administratori mogu ograničiti brzinu kojom primarni čvor primenjuje operacije sa ciljem da ciljem da se zadrži urezano kašnjeje ispod konfigurabilne maksimalne vrednosti *flowControlTargetLagSeconds*.

## Write concern

Write concern za skupove replika predstavljaju broj članova koji nose podatke, odnosno broj primarnih i sekundarnih čvorova, ali arbitražnih čvorova ne, i koji moraju da potrvde operaciju pisanja kako bi ona bila uspešna. Operacija pisanja može da se potvrdi tek nakon što je član primio i uspešno primenio upis.

Ako je postavljeno parametar *w* postavljen na *w: "majority"* to znači da write concern zahteva potvrdu da su operacije pisanja trajno izvršene na većini članova sa pravom glasa. Ovo je podrazumevana vrednost za većinu konfiguracija. U slučaju da je parametar postavljen na *w: 1* zahteva se smao potrvda od primarnog člana pre vraćanja potrvde. Write concern sa numeričkom vrednošću koja je veća od 1 zahteva potvrdu od primarnog čvora i onoliko sekundarnih koliko je potrebno da se upotpuni navedena vrednost. Navedena vrednost mora da bude manja ili jednaka ukupnom broju članova koji u sebi imaju podatke.



Slika 7 – Write concern

Na slici 7 je prikazan proces pisanja i potvrda većine. Aplikacija izdaje operaciju pisanja i čeka dok primarna ne primi potvrdu od potrebnog broja članova. Ako je broj potrebih potvrda veće od 1 ili većina, primarni čvor čeka dok potreban broj sekundarnih čvorova ne potvrdi upis. Ako je potrebno da samo jedan čvor potrvdi upis, odnosno ako je w: 1, primarni čvor može da vrati potvrdu pisanja čim lokalno primeni upis. Što je više članova koji priznaju upis, manja je verovatnoća da će se pisani podaci vratiti nazad ako primarni ne uspe. Što je veći broj potrvda koje se očekuju, to je kašnjeje veće jer je potrebno da se čeka dok se ne dobije traženi nivo potrvde. Idealna vrednost za write concern zavisi od ciljeva i performansi aplikacija i zahteva.

Aplikacija čeka dok nije potrvđena promarna povratna informacija. Na primer, ako postoji skup replika od 3 člana, jedan primarni i dva sekundarna, operacija bi zahtevala potvrdu 2 od 3 člana. Ako se kasnije skup skalira na 5 članova, operacija bi zahtevala 3 potvrde. Ako primarni čvor ne vrati potvrdu unutar vremenskog ograničenja, operacija pisanja ne uspeva. Ako istekne vreme čekanja ukazuje se na to da operaciju pisanja nije potvrdio potreban broj čvorova u određenom vremenskom periodu i to ne znači nužno da primarni čvor nije uspešno izvršio operaciju.

Na slici 8 je prikazano navođenje vrednosti za wirte concern, kao i vrmemensko ograničenje za koje aplikacija čeka.

A picture containing text, font, screenshot, line

Description automatically generated

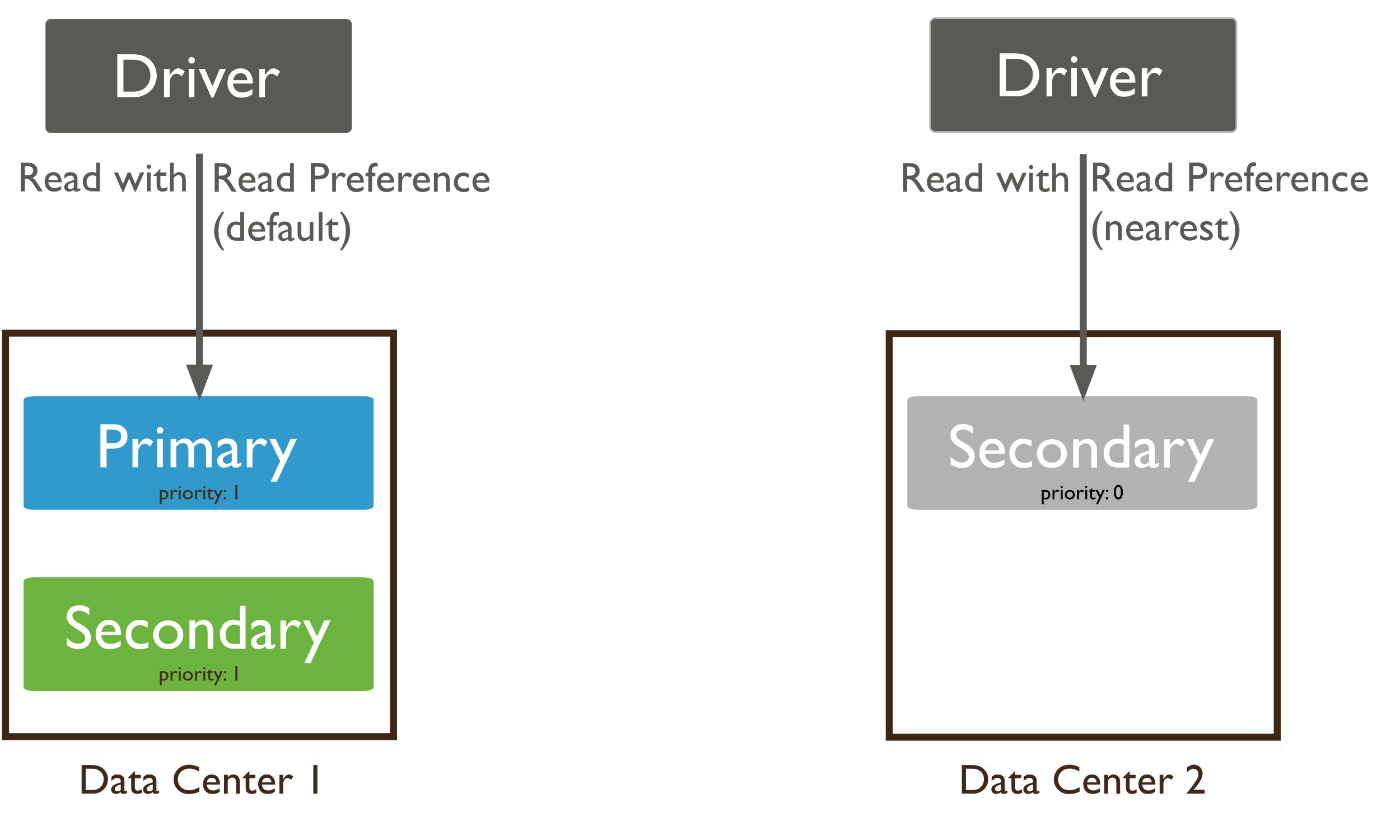
Slika 8 – postavljanje parametara

## Read preference

Read preference (preferencija čitanja) opisuje kako MongoDB klijenti usmeravaju operacije čitanja do članova skupa replika. Podrazumevano je da aplikacijaa usmerava svoje operacije čitanja primarnom članu. Međutim, moguće je da se navede opcija čitanja. Preferencija za čitanje se sastoji od režima preferencije čitanja i liste skupova oznaka, opciono.

Postoji nekoliko modova:

* **primary** – sve operacije čitanja koriste samo trenutni primarni skup replika i ovo je podrazumevani režim čitanja. U slučaju da primarni čvor nije dostupan, dolazi do greške prilikom operacije čitanja ili do izuzetka.
* **primaryPreferred** – u većini slučajeva, operacije čitanja se obavlja iz primarnog čvora. Međutim, ako je primarni čvor nedostupan, operacija čitanja se obavlja sa sekundarnih čvorova. Kada preferencija čitanja uključuje listu oznaka i ne postoji primarni čvor, klijent pokušava da pronađe sekundarne članove sa odgovarajućim oznakama. Ako se pronađu podudarni sekundari, klijent nasumično bira sekundar iz najbliže grupe. Ako nijedan sekundar nema odgovarajuću oznaku, dolazi do greške. Čitanje pomoću ovog režima može da vrati zastarele podatke.
* **secondary** – operacije čitanja se obavljaju samo na sekundarnim čvorovima i ukoliko oni nisu dostupni dolazi do greške prilikom ove operacije. Kada uključuje listu oznaka, klijent pokušava da pronađe sekundarne članove sa odgovarajućim. Ako se pronađu podudarni sekundari, klijent bira nasumični sekundar iz najbliže grupe odgovarajućih sekundarnih podataka.
* **secondaryPreferred** – u većini slučajeva čitanje se obavlja iz sekundarnih replika, ali u ako u skupu replika postoji samo primarni član onda će se čitanje obaviti iz primarnog.
* **nearest** – čitanje se obavlja sa člana čija je mrežna latencija najprihvatljivija, bez obzira na to da li je to primarni ili sekundarni čvor, u ovom režimu su oni jednaki. Pomoću ovog režima minimizira se kašnjenje.



Slika 9 – Read preference

# Zaključak

Replikacija podataka osigurava visoku dostupnost podataka. Kroz koncept skupa replika, MongoDB omogućava da se primarni čvor automatski zameni sekundarnim čvorom u slučaju neuspeha ili prekida rada. To osigurava kontinuiranu dostupnost podataka i smanjuje rizik od gubitka ili nedostupnosti podataka. Takođe, replikacija podataka u MongoDB-u pruža horizontalnu skalabilnost i bolju raspodelu opterećenja. Dodavanje sekundarnih čvorova pruža raspodelu opterećenja između više čvorova, čime se poboljšavaju. Omogućava brže čitanje podataka. Sekundarni čvorovi mogu se koristiti za čitanje podataka, što daje bolje performanse pristupa podacima. Ovo je korisno u situacijama gde se često čita velika količina podataka, jer se opterećenje raspređuje na više čvorova.

Kada se sve ovo uzme u obzir, može se zaključiti da je replikacija ključna stvar za održanje pouzdanosti, skalabilnosti i dostupnosti podataka. Upravljanje replikacijom podataka u MongoDB-u zahteva pažljivo planiranje i konfiguraciju kako bi se osigurala doslednost podataka, minimizirala latencija replikacije i rukovalo konfliktima.