



Univerzitet u Nišu
Elektronski fakultet
Katedra za računarstvo



Visoka dostupnost kod MongoDB baze podataka

Mentor:

Prof. dr Aleksandar Stanimirović

Student:

Draginja Anđelković, 1028

Niš, septembar 2021.

Sadržaj

Sadržaj.....	2
1. Uvod.....	4
2. Kvarovi i ograničenja kod MongoDB baze podataka	6
2.1. Ograničenja 32-bitne verzije MongoDB baze podataka	6
2.2. Otkazivanje u skupu replika	6
2.3. Distribucija prilikom kvara konfiguracionog servera	7
2.4. Zaključavanje baze podataka	7
3. Klasterovanje MongoDB baze podataka	8
3.1. Skup replika	8
3.2. Vertikalno i horizontalno skaliranje	9
3.3. Implementacija klastera u MongoDB bazi podataka	11
4. Korišćenje skupa replika	12
4.1. Replikacija.....	12
4.2. Arhitektura skupa replika	12
4.3. Uloga dnevnika operacija u replikaciji	13
4.4. Praktičan primer postavljanja skupa replika	13
4.4.1. Konfiguracija skupa replika.....	14
4.4.2. Rapoređivanje skupa replika	15
4.4.3. Dodavanje novog čvora	16
4.4.4. Uklanjanje člana	18
4.4.5. Dodavanje arbitra	18
4.4.6. Testiranje metoda zaštite sistema od kvara.....	19
5. Koncept deljenja baze podataka	22
5.1. Skaliranje	22
5.2. Deljenje	22
5.2.1. Moduli podele	23
5.2.2. Korišćenje ključeva za deljenje	23

5.3. Proces cepanja i uravnoteženja	24
5.4. Praktičan primer deljenja baze podataka	24
5.4.1. Implementacija konfiguracionih servera	25
5.4.2. Konfigurisanje rutera za upite (mongos instance)	26
5.4.3. Upravljanje mongos instancom	26
5.4.4. Dodavanje delova mongos procesu	27
5.4.5. Omogućavanje deljenja	28
6. Analiza i poboljšanje performansi baze podataka	31
6.1. Profilisanje	31
6.1.1. Korišćenje profilisanja	31
6.1.2. Omogućavanje i konfigurisanje profila	32
6.2. Druge analitičke metode	33
6.3. Uvođenje indeksa	33
6.4. Korišćenje projekcije	34
6.5. Ograničavanje broja vraćenih zapisa	34
6.6. Poboljšanja na nivou hardvera	34
7. Migracija instanci i smanjenje zastoja	35
8. Nadgledanje baze podataka i rešavanje problema nad bazom podataka	36
8.1. Korišćenje funkcije profilisanja	36
8.2. Korišćenje ugrađenih alata za izveštavanje	37
8.3. Korišćenje uslužnih programa zasnovanih na internetu	37
9. Zaključak	38
10. Literatura	39

1.Uvod

Visoka dostupnost (engl. *high availability*) je sposobnost sistema da radi neprekidno, bez greške u određenom vremenskom periodu.

Za razliku od prošlosti kada su korisnici očekivali da će infrastruktura s vremena na vreme otkazati, zastoji su danas neprihvatljivi u većini konteksta. U doba kada usluga koja ne radi samo nekoliko sati može značajno uticati na sposobnost preduzeća da održava poslovanje, održavanje visoke dostupnosti je ključno. [2]

Nemoguće je da sistemi budu dostupni 100% vremena, pa pravi sistemi visoke dostupnosti uglavnom teže pet devetki standardu operativnih performansi, što znači da su sistem ili proizvod dostupni 99,999% vremena. [3]

U većini slučajeva, kada ljudi govore o visokoj dostupnosti, razmišljaju o aplikacijama i uslugama. Ipak, koncept se može i treba proširiti na podatke. Na kraju krajeva, bez podataka aplikacije i usluge nisu baš korisne. Ako planirate strategiju visoke dostupnosti koja se odnosi samo na aplikacije, a ne i na podatke, nećete uspeti da obezbedite potupni kontinuitet poslovanja.

Sledeća razmatranja treba da utiču na strategiju visoke dostupnosti podataka [3]:

- Serveri koji hostuju podatke moraju biti otporni na smetnje. To možete postići korišćenjem redundantnih servera za hostovanje vaših podataka i/ili automatskim prelaskom prilikom kvara.
- Baze podataka treba da budu modelovane tako da otkaz jednog čvora baze podataka ne dovede do toga da baza podataka bude nedostupna. Baze podataka bi takođe trebale biti u mogućnosti da se automatski ponovo pokrenu ako se sruše, kako bi se smanjili zastoji.
- Ako se oslanjate na mrežu za pristup podacima, dostupost mreže je važna komponenta.

Visoka dostupnost podataka može ići i korak dalje. Visoko dostupni podaci su podaci sa kojima možete lako da radite. To su kvalitetni podaci koji su dostupni u formatima koji su vam potrebni da biste radili sa njima. To su podaci koji su kompatibilni sa bilo kojim alatom koji koristite za analizu i tumačenje. [2]

Poglavlje 1 ovog rada je uvod u obrađenu temu, Visoku dostupnost kod MongoDB baze podataka.

Poglavlje 2, Kvarovi i ograničenja MongoDB baze podataka, pokriva greške koje se mogu javiti kod MongoDB baze podataka, poput zastoja servera, grešaka pri upisu ili čitanju podataka i tako dalje. Ovo poglavlje takođe razmatra rešenja za smanjenje zastoja.

U okviru poglavlja 3, Klasterovanje MongoDB baze podataka dat je pregled MongoDB klaster rešenja u proizvodnom okruženju.

Prvi deo poglavlja 4, Korišćenje skupa replika, pokriva skupove replika i objašnjava osnovne koncepte skupova replika. U nastavku poglavlja dati su primeri iz stvarnog sveta, kako bi se pomoću skupova replika obezbedila visoka dostupnost servera.

Poglavlje 5, Koncept deljenja baze podataka, objašnjava upotrebu funkcije deljenja u MongoDB bazi podataka. U nastavku poglavlja dat je praktičan primer deljenja baze podataka, kao i omogućavanja klasterovanja u postojećoj bazi podataka.

Poglavlje 6, Analiza i poboljšanje performansi baze podataka, pokriva upotrebu najnovijih MongoDB funkcija za poboljšanje performansi čitanja i pisanja.

U okviru poglavlja 7, Migracija instanci i smanjenje zastoja obrađena je migracija baze podataka i ponuđena su rešenja za smanjenje zastoja servera do koga dolazi prilikom migracije baze podataka.

Poglavlje 8, Nadgledanje baze podataka i rešavanje problema nad bazom podataka, obrađuje alate i tehnike za upravljanje performansama baze podataka. Ovo poglavlje takođe nudi načine rešavanja problema nad bazom podataka.

U okviru poslednjeg poglavlja ovog rada, poglavlja 9, navedeni su najznačajniji zaključci obrađene teme.

2. Kvarovi i ograničenja kod MongoDB baze podataka

Programeri baza podataka i mašine (engl. *engine*) se mogu suočiti sa gubitkom podataka ili greškama tokom rada sa serverom. Ovaj rad ima za cilj da pruži rešenja za uvek dostupan mehanizam MongoDB baze podataka. Za početak se treba upoznati sa glavnim razlozima neuspeha i ograničenjima MongoDB baze podataka.

2.1. Ograničenja 32-bitne verzije MongoDB baze podataka

Ako se MongoDB primenjuje u proizvodnom okruženju, neophodno je koristiti 64-bitnu verziju, a ne 32-bitnu. U 32-bitnoj verziji, MongoDB ima ograničenje veličine memorije, pa se ne mogu čuvati skupovi podataka veći od 2 GB. Ako je skladište baze podataka veće od 2 GB, dobija se greška i nije moguće pokrenuti server sve dok se podaci ne uklone ili baza podataka ne migrira na 64-bitnu verziju MongoDB baze podataka.

2.2. Otkazivanje u skupu replika

Skup replika u MongoDB bazi podataka je grupa mongod procesa koji održavaju isti skup podataka. Skupovi replika pružaju redundantnost i visoku dostupnost podataka.

U skupovima replika, kada primarni čvor postane nedostupan, skup replika sprovodi izborni postupak da bi se izabrao novi primarni čvor iz sekundarnih. Ova promena može trajati oko 10 do 60 sekundi. Međutim, ona se razlikuje između servera i zavisi od postavki skupa replika, broja čvorova i kapaciteta servera.

Tokom izvođenja izbornog procesa skup replika postaje nedostupan za operacije pisanja.

U poglavlju 4 će biti više reči o visokoj dostupnosti skupova replika.

2.3. Distribucija prilikom kvara konfiguracionog servera

Za distribuciju celog skupa podataka na različite mašine i servere, odnosno horizontalno skaliranje, MongoDB pruža funkciju deljenja (engl. *sharding*). Korišćenjem ove funkcije moguće je razdvojiti bazu podataka na različite delove na odvojenim serverima.

MongoDB koristi konfiguracione servere za čuvanje konfiguracije za podešavanje deljenja i čuvanje odnosa između članova.

U proizvodnom okruženju postoje tačno tri konfiguraciona servera. Ako jedan ili dva postanu nedostupni, baza podataka se može koristiti samo za čitanje. Ako sva tri konfiguraciona servera postanu nedostupna, klaster će takođe biti nedostupan. U tom slučaju bi trebalo zameniti konfiguracione servere što je pre moguće.

Komadići (engl. *chunks*) su delovi baze podataka na odvojenim mašinama, a MongoDB cepa bazu podataka na delove na osnovu ključa za deljenje.

O funkciji deljenja baze i ključevima za deljenje će biti više reči u poglavlju 5.2.2.

2.4. Zaključavanje baze podataka

MongoDB koristi brave za čitanje/pisanje kako bi sprečilo sukobe između operacija čitanja/pisanja, a takođe osigurava da klijenti istim upitom dobiju isti skup podataka.

MongoDB koristi zajedničku bravu za čitanje, što znači da mnoge operacije mogu koristiti postojeću bravu za čitanje. S druge strane, upis je ograničen na jednu operaciju, što znači da nijedna druga operacija upisivanja ne može koristiti postojeću bravu.

Brave kod MongoDB baze podataka su “pohlepne za pisanjem”, što znači da ako obe operacije, i čitanja i pisanja čekaju zaključavanje, operacija pisanja dobija veći prioritet, pa MongoDB zaključava prvo operaciju pisanja.

3. Klasterovanje MongoDB baze podataka

Klasterovanje je tehnika grupisanja podataka u grupe (klaster) na osnovu sličnih osobina, čineći ih preglednijim i korisnijim. [4]

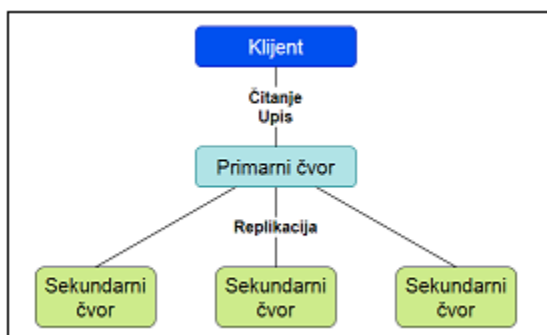
Kako se baza podataka povećava, mora se doći do rešenja za poboljšanje protoka i njenih performansi kako bi više upita bilo podržano tokom određenog vremenskog perioda. Najčešće rešenje u bazama podataka je klasterovanje. Korišćenjem klastera podaci se čuvaju na više servera i na taj način se pritisak na bazu distribuira između različitih servera. U ovom poglavlju će biti više reči o klasterovanju u MongoDB bazi podataka.

3.1. Skup replika

Skup replika je kolekcija mongod¹ procesa, od kojih je jedan primarni mongod proces koji prihvata sve operacije pisanja i čitanja od klijenta. Svi ostali procesi su sekundarni.

Postupak sinhronizacije podataka između primarnog i sekundarnih čvorova se dešava uz pomoć replikacije. Da bi podržao replikaciju, primarni čvor evidentira sve operacije upisa u svoj dnevnik operacija (engl. *oplog*).

Slika 1 ilustruje odnose između klijenta, primarnog i sekundarnih čvorova.



Slika 1: Odnos između klijenta, primarnog i sekundarnih čvorova. Proces replikacije

¹ Mongod je primarni proces koji je MongoDB bazi podataka potreban za pokretanje servera. Upravlja zahtevima ili upitima i rukuje konekcijama.

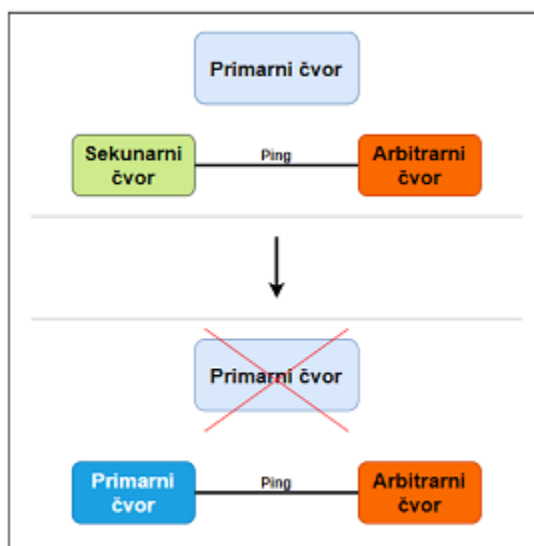
Da bi ažurirao dostupnost čvorova, svaki čvor šalje ping pakete ili pakete otkucaja srca (engl. *heartbeat*) svim ostalim čvorovima.

Ako je primarni čvor nedostupan, tj. ne primi ili ne pošalje ping poruku u roku od 10 sekundi, skup replika pokreće se izborni proces kako bi se od sekundarnih izabrao novi primarni čvor. Sekundarni čvor će biti izabran za primarni ako postoji čista većina glasova za taj čvor. U izbornom procesu pored sekundarnih čvorova može učestvovati i arbitarski čvor.

Arbitarski čvorovi nemaju nijedan skup podataka i ne mogu postati primarni čvorovi. Oni postoje samo za glasanje u izbornom procesu. Od koristi su kada u imamo paran broj čvorova.

Kada se stari primarni čvor ponovo poveže sa mrežom postaće novi sekundarni čvor.

Na slici 2 je prikazan izborni proces prilikom izbora novog primarnog čvora.

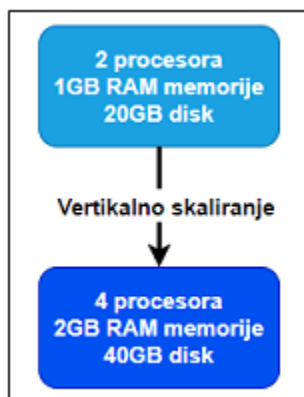


Slika 2: Izborni proces

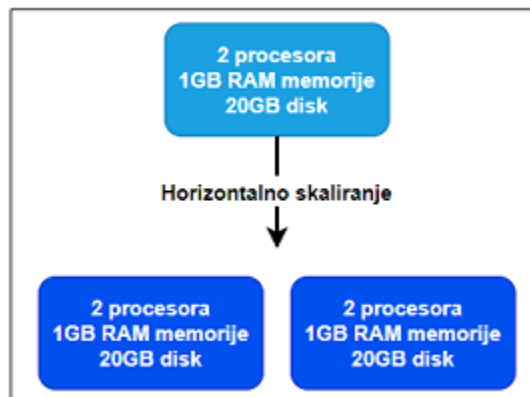
3.2. Vertikalno i horizontalno skaliranje

Sistemi baza podataka se mogu skalirati na dva načina, vertikalno i horizontalno. Kod vertikalnog skaliranja potrebno je dodati veći kapacitet postojećem serveru, na primer, dodajući više CPU-a, RAM-a ili memorije serveru. Dodavanje više procesora ili RAM-a na postojeći server je skuplje i teže za održavanje. Postupak vertikalnog skaliranja prikazan je na slici 3.

Kod horizontalnog skaliranja, sistem baze podataka se može skalirati kroz različite servere, tj. horizontalno. Ovaj pristup je sigurniji, lakši i jeftiniji od vertikalnog skaliranja. Proces horizontalnog skaliranja prikazan je na slici 4.

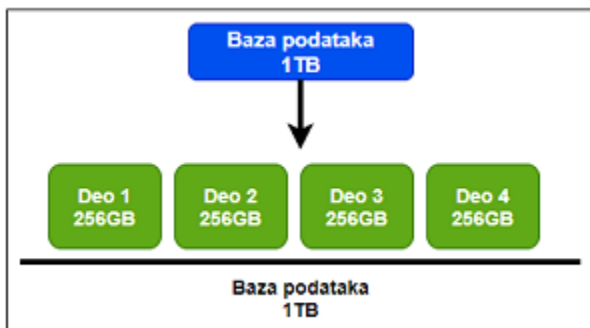


Slika 3: Vertikalno skaliranje



Slika 4: Horizontalno skaliranje

Horizontalno skaliranje je u MongoDB bazi podataka implementirano pomoću deljenja (engl. *sharding*), pri čemu je isti skup podataka podeljen na više servera, gde je svaki pojedinačni server nezavisna baza podataka. Svi delovi (engl. *shard*) zajedno čine jedinstvenu logičku bazu podataka. Primer podele baze podataka od 1TB na četiri različita servera od kojih svaki ima memoriju od 256GB prikazan je na slici 5. Inače bi skladištenje tako velikog skupa podataka na jednom serveru bio bi veliki izazov.



Slika 5: Podela baze podataka

Deljenje rešava problem preopterećenja i velikog skupa podataka. Tokom upisivanja ili čitanja podataka, klijent ili aplikacija pristupa samo jednom delu, čime se protok i opterećenje takođe horizontalno skaliraju. Time baza podataka može podržati više operacija.

3.3. Implementacija klastera u MongoDB bazi podataka

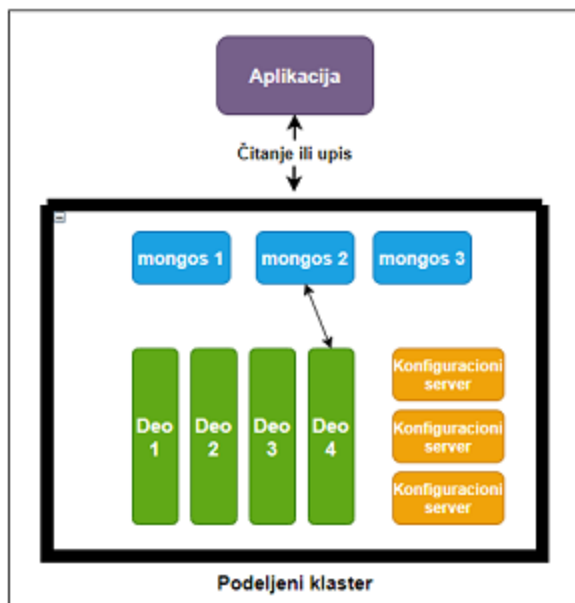
Podjeljeni klaster (engl. *sharded cluster*) u MongoDB bazi podataka se sastoji od tri komponente, konfiguracionih servera, delova (skupova replika) i rutera upita za sve klijente.

Konfiguracioni serveri se koriste za skladištenje mreže klastera i metapodataka svakog čvora.

Delovi su skupovi replika koji čuvaju svaki pojedinačni deo baze podataka.

Ruter za upite je interfejs za klijenta i aplikacije. Aplikacija šalje zahtev ruteru za upite, a ruter za upite usmerava zahtev na odgovarajuću repliku, a po završetku operacije rezultat vraća klijentu. Svrha rutera za upite je podela opterećenja između različitih delova. Klaster može sadržati jedan ili više rutera za upite.

Veza između različitih komponenti izdijeljenog klastera u proizvodnom okruženju prikazana je na slici 6.



Slika 6: Odnos između različitih komponenti izdijeljenog klastera

Podjeljeni klaster koristi ključ za deljenje za distribuciju skupa podataka kroz delove ili deljenje baze podataka. Postoje dva različita pristupa za odabir ključa za deljenje: particionisanje zasnovano na opsegu i particionisanje zasnovano na hešu. Više reči o njima biće u poglavlju 5.2.2.

4. Korišćenje skupa replika

U ovom poglavlju će biti reči o replikaciji i skupu replika, zajedno sa nekim praktičnim primerima.

4.1. Replikacija

Replikacija je proces sinhronizacije podataka koji se nalaze na više servera. [5]

U skupu replika, svaki čvor može biti hostovan na različitim serverima. Zbog toga je neophodna sinhronizacija podataka između čvorova.

Postupak replikacije koji se koristi za sinhronizaciju podataka između primarnog čvora, koji prihvata sve operacije čitanja/pisanja, i sekundarnih čvorova vidi se na slici 1. Nakon završetka procesa sinhronizacije svi sekundarni čvorovi i primarni čvor će imati isti skup podataka.

Uz postupak replikacije postoji višak kopija podataka u bazi, ali taj višak doprinosi dostupnosti podataka, štiti bazu podataka od kvarova, gubitka podataka ili padova hardvera, i povećava performanse čitanja pri velikom opterećenju.

Proces replikacije može biti i asinhroni.

4.2. Arhitektura skupa replika

Svaki skup replika se sastoji od različitih članova od kojih je svaki odgovoran za određeni zadatak. Sledeća lista prikazuje sve moguće uloge za svakog člana:

- **Primarni:** Primarni čvor je odgovoran za prihvatanje svih operacija čitanja/pisanja od klijenta. Svaki skup replika ima tačno jedan primarni čvor. Pod nekim uslovima primarni čvor može postati sekundarni ili sekundarni može postati primarni.
- **Sekundarni:** Sekundarni čvorovi imaju isti skup podataka kao i primarni. Svaki skup replika može imati jedan ili više sekundarnih čvorova. Sekundarni čvor u izbornom postupku može postati primarni.

- **Arbitar:** Čvor arbitar se koristi samo za glasanje u izbornom postupku. Ovaj čvor ne menja svoje stanje i nije domaćin nijednom skupu podataka.

Svaki skup replika može imati do 12 članova, od kojih 7 može glasati na izborima.

4.3. Uloga dnevnika operacija u replikaciji

Dnevnik operacija (engl. *oplog*) je posebna ograničena kolekcija koja čuva tekući zapis svih operacija koje modifikuju podatke uskladištene u bazama podataka. [6]

To je kolekcija fiksne veličine i automatski će prebrisati stare zapise kada kolekcija dostigne maksimalnu veličinu. Podrazumevana veličina dnevnika operacija kod MongoDB baze podataka se može promeniti pomoću svojstva `oplogSize`.

Dnevnik operacija se koristi u procesu asinhronne replikacije. Naime, klijenti šalju operacije čitanja/pisanja na primarni čvor, a on primenjuje operacije i upisuje ih u svoj dnevnik operacija. Sekundarni članovi kopiraju i primenjuju dnevnik operacija primarnog čvora kako bi imali ažurne kopije podataka.

4.4. Praktičan primer postavljanja skupa replika

U okviru praktičnog dela ovog poglavlja postavljen je novi skup replika od nule kako bi se obezbedio lako dostupan MongoDB server. Takođe, ispraćen je proces otkazivanja u skupu replika.

Postoje dva načina za postavljanje skupa replika, za proizvodnju ili razvoj i testiranje. Ovde ćemo postaviti skup replika za proizvodnu upotrebu.

Tročlani skup replika je dovoljno snažan da prevaziđe probleme sa mrežom i serverom. Takođe, za nesmetan izborni proces, preporučuje se da skup replika ima neparan broj članova.

Za proizvodnu upotrebu skupa replika, svaki skup replika treba da bude smešten na različitim mašinama, da se podaci ne izgube ukoliko mašina otkaže.

Pre postavljanja skupa replika, treba se uveriti da je na svakoj mašini instalirana najnovija stabilna verzija MongoDB baze podataka. Takođe, treba se uveriti da između mašina gde su

smeštene replike nema bilo kakvih mrežnih problema, tj. da svaka mašina vidi drugu i da je zaštitni zid (engl. *firewall*) konfigurisan tako da ne sprečava veze.

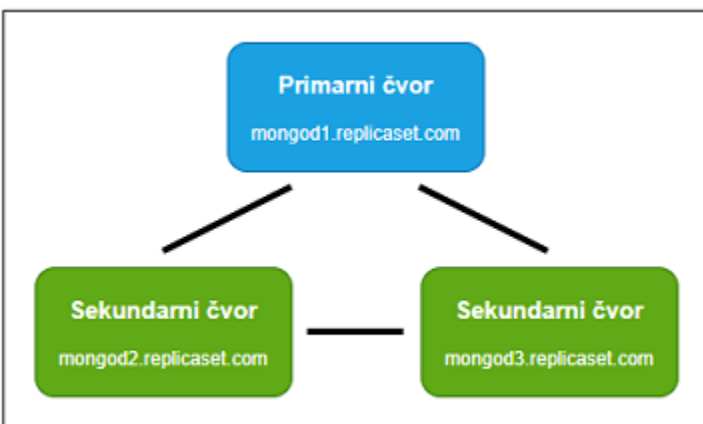
4.4.1. Konfiguracija skupa replika

Primer predstavlja skup replika sa tri člana na tri odvojene mašine. Na glavnom računaru je Mac OS X, a mrežne veze između članova i pokretanje drugih mašina odjednom obezbeđuje VMware Fusion.

Ime hosta svake mašine je:

- mongod1.replicaset.com
- mongod2.replicaset.com
- mongod3.replicaset.com

Odnos između članova skupa replika prikazan je na slici 7.



Slika 7: Odnos između članova skupa replika

Sledeći korak je priprema konfiguracione datoteke koja je neophodna za pokretanje mongod instance u proizvodnom okruženju. Za pokretanje mongod ili mongos pomoću konfiguracione datoteke, mogu se koristiti sledeće naredbe:

```
mongod --config /etc/mongod.conf mongod -f /etc/mongod.conf mongos --config /srv/mongod/mongos.conf mongos -f /srv/mongod/mongos.conf
```

U ovom primeru korišćena je putanja `/etc/mongod.conf` za skladištenje konfiguracione datoteke. Sledeća naredba prikazuje sadržaj ove datoteke:

`dbpath = /usr/local/var/mongodb logpath = /var/log/mongodb.log bind_ip = 127.0.0.1 port = 27017 fork = true replSet = rs1`

Ukratko ćemo razmotriti neke od važnih opcija. Za definisanje putanje uskladištenih podataka koristi se svojstvo *dbpath*. Svojstvo *logpath* definiše putanju datoteke dnevnika. Pomoću naredbe *bind_ip* definiše se mreži interfejs koji mongod ili mongos sluša. Svojstvo *port* definiše broj porta koji proces sluša. Podrazumevani broj porta za mongod proces je 27017. Svojstvo *fork* služi za pokretanje mongod ili mongos procesa kao demona. Svojstvo *replSet* se koristi za definisanje imena skupa replika.

Nakon čuvanja konfiguracione datoteke, mongod instanca se može pokrenuti korišćenjem – config parametra.

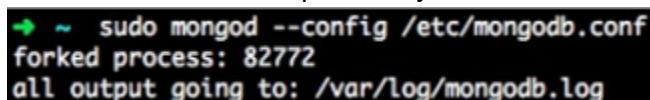
4.4.2. Rapoređivanje skupa replika

Nakon inicijalizacije može se definisati skup replika. Prvo je potrebno pokrenuti mongod proces na svim mašinama kroz sledeće korake:

1. Neophodno je imati konfiguracionu datoteku na svom uređaju, a zatim pokrenuti sledeću komandu:

`mongod --config /etc/mongodb.conf`

Rezultat ove komande prikazan je na slici 8.



```
➔ ~ sudo mongod --config /etc/mongodb.conf
forked process: 82772
all output going to: /var/log/mongodb.log
```

Slika 8

2. S obzirom da smo fork svojstvo postavili na true, mongod se pokreće kao demon u pozadini, a izlazi bi trebalo da se vide u datoteci dnevnika. Sadržaj ove datoteke prikazan je na slici 9.



```
➔ ~ cat /var/log/mongodb.log
Thu Aug 26 17:58:51 [initandlisten] MongoDB starting : pid=42855 port=27017 dbpath=/usr/local/var/mongodb 64-bit host
-mongod1.replicaset.com
Thu Aug 26 17:58:51 [initandlisten] db version v2.6.0, point version 4.3
Thu Aug 26 17:58:51 [initandlisten] git version: 695c67d7f7f361b4564e13366f877ca0404222
Thu Aug 26 17:58:51 [initandlisten] build info: Darwin armv2.10gen.cc 9.6.0 Darwin Kernel Version 9.6.0: Mon Nov 24 17
:37:40 PST 2008; root:xnu-1228.9.59~1/RELEASE_ARM_T3040 ARMV2T.LTS_VERSION=1.40
Thu Aug 26 17:58:51 [initandlisten] options: { bind_ip: "127.0.0.1", config: "/etc/mongodb.conf", dbpath: "/usr/local
/var/mongodb", fork: "true", logpath: "/var/log/mongodb.log", port: 27017, replSet: "rs1" }
Thu Aug 26 17:58:51 [initandlisten] journal dir=/usr/local/var/mongodb/journal
Thu Aug 26 17:58:51 [initandlisten] recover : no journal files present, no recovery needed
Thu Aug 26 17:58:51 [webserver] admin web console waiting for connections on port 28017
Thu Aug 26 17:58:51 [initandlisten] waiting for connections on port 27017
Thu Aug 26 17:58:51 [initandlisten] connection accepted from 127.0.0.1:61874 #1
Thu Aug 26 17:58:51 [rsstart] replset can't get local.system.replset config from self or any seed (EMPTYCONFIG)
Thu Aug 26 17:58:51 [rsstart] replset info you may need to run replSetInitiate -- rs.initiate() in the shell -- if th
at is not already done
```

Slika 9

3. Za sledeće korake treba koristiti mongo interaktivnu ljusku za pokretanje i konfigurisanje skupa replika.

4. Pokretanje mongo komande iz komandnog okruženja prikazano je na slici 10.

```
→ /etc mongo
MongoDB shell version: 2.6.0
connecting to: test
```

Slika 10

5. Naredba `rs.initiate()` u mongo procesu pokreće skup replika.
6. Nakon izvršavanja prethodne naredbe dobija se rezultat kao na slici 11.

```
> rs.initiate()
{
  "info2" : "no configuration explicitly specified -- making one",
  "me" : "mongod1.replicaset.com:27017",
  "info" : "Config now saved locally. Should come online in about a minute.",
  "ok" : 1
}
```

Slika 11

Ako je inicijalizacija skupa replika uspešna dobija se poruka ok: 1 iz mongo procesa.

Korišćenjem `rs.config()` komande u mongo procesu, dobija se konfiguracija skupa replika koja je prikazana na slici 12. Svojstvo `_id` prikazuje ime skupa replika, a trenutno se u skupu replika nalazi samo jedan član, primarni čvor. Svojstvo `host` prikazuje adresu i port svakog člana. Host našeg primarnog čvora je `mongod1.replicaset.com`, a broj porta je 27017.

```
PRIMARY> rs.conf()
{
  "_id" : "rs1",
  "version" : 1,
  "members" : [
    {
      "_id" : 0,
      "host" : "mongod1.replicaset.com:27017"
    }
  ]
}
```

Slika 12

4.4.3. Dodavanje novog čvora

Nakon kreiranja primarnog čvora, mogu mu se dodati drugi članovi, uključujući sekundarne čvorove i arbitre.

Do sada je konfigurisan `mongod1.replicaset.com`, a sada treba dodati `mongod2.replicaset.com` i `mnogod3.replicaset.com`. Ovaj postupak je naveden u sledećim koracima:

1. Konfiguraciona datoteka treba da bude iskopirana na ciljnoj mašini, i da se mongod proces pokrene koristeći `-config` parametar.
2. Dodavanje novog člana u postojeći skup replika se obavlja pozivanjem sledeće naredbe u mongo ljusci:
`rs.add("mongod2.replicaset.com")`
3. Nakon izdavanje prethodne komande dobija se rezultat prikazan na slici 13.


```
PRIMARY> rs.add("mongod2.replicaset.com")
{ "ok" : 1 }
```

Slika 13

4. Izvršavanjem naredbe `rs.status()` moguće je videti status selog skupa replika u svakom trenutku.
5. Zatim se skupu replika dodaje treći član pomoću metode `rs.add()`. Ime domaćina trećeg člana je `mongod3.replicaset.com`. Dodavanje trećeg člana prikazano je na slici 14.

```
{
  "set" : "rs1",
  "date" : ISODate("2021-08-11T15:58:44Z"),
  "myState" : 1,
  "members" : [
    {
      "_id" : 0,
      "name" : "mongod1.replicaset.com:27017",
      "health" : 1,
      "state" : 1,
      "stateStr" : "PRIMARY",
      "uptime" : {
        "s" : 13972318000,
        "i" : 1
      },
      "optimeDate" : ISODate("2021-08-11T15:58:38Z"),
      "self" : true
    },
    {
      "_id" : 1,
      "name" : "mongod2.replicaset.com:27017",
      "health" : 1,
      "state" : 2,
      "stateStr" : "SECONDARY",
      "uptime" : 45,
      "uptime" : {
        "s" : 13972318000,
        "i" : 1
      },
      "optimeDate" : ISODate("2021-08-11T15:58:38Z"),
      "lastHeartbeat" : ISODate("2021-08-11T15:58:43Z"),
      "pingMs" : 0
    },
    {
      "_id" : 2,
      "name" : "mongod3.replicaset.com:27017",
      "health" : 1,
      "state" : 6,
      "stateStr" : "UNKNOWN",
      "uptime" : 0,
      "uptime" : {
        "s" : 0,
        "i" : 0
      },
      "optimeDate" : ISODate("1970-01-01T00:00:00Z"),
      "lastHeartbeat" : ISODate("2021-08-11T15:58:42Z"),
      "pingMs" : 0,
      "errmsg" : "still initializing"
    }
  ],
  "ok" : 1
}
```

Slika 14

6. Nakon dodavanje trećeg člana izdavanjem naredbe `rs.status()` u mongo ljusci može se proveriti status skupa replika.
7. Nakon nekog vremena `mongod2.replicaset.com` postaje primarni član. U mongo ljusci sekundarne mašine vidi se rezultat prikazan na slici 15.

```
rs1:RECOVERING>
rs1:SECONDARY>
```

Slika 15

8. Mongod objekat prelazi u režim oporavka i sinhronizuje podatke sa primarnim čvorom.
9. Nakon sinhronizacije podataka između primarnog i sekundarnog čvora, sa primarnog čvora se može videti status kao što je prikazano na slici 16.

```
PRIMARY> rs.status()
{
  "set" : "rs1",
  "date" : ISODate("2011-08-11T13:38:26Z"),
  "myState" : 1,
  "members" : [
    {
      "_id" : 0,
      "name" : "mongod2.replicaset.com:27017",
      "host" : 1,
      "state" : 1,
      "stateStr" : "PRIMARY",
      "uptime" : {
        "u" : 186723318000,
        "s" : 1
      },
      "optimeDate" : ISODate("2011-08-11T13:38:26Z"),
      "self" : true
    },
    {
      "_id" : 1,
      "name" : "mongod2.replicaset.com:27017",
      "host" : 1,
      "state" : 2,
      "stateStr" : "SECONDARY",
      "uptime" : {
        "u" : 186723318000,
        "s" : 1
      },
      "optimeDate" : ISODate("2011-08-11T13:38:26Z"),
      "lastHeartbeat" : ISODate("2011-08-11T13:38:26Z"),
      "pingMs" : 0
    },
    {
      "_id" : 2,
      "name" : "mongod3.replicaset.com:27017",
      "host" : 1,
      "state" : 2,
      "stateStr" : "SECONDARY",
      "uptime" : {
        "u" : 186723318000,
        "s" : 1
      },
      "optimeDate" : ISODate("2011-08-11T13:38:26Z"),
      "lastHeartbeat" : ISODate("2011-08-11T13:38:26Z"),
      "pingMs" : 0
    }
  ]
}
```

Slika 16

4.4.4. Uklanjanje člana

Da biste uklonili člana iz postojećeg skupa replika, u metod uklanjanja treba uvesti ime hosta i broj porta. Nakon izvršenja naredbe za uklanjanje čvora dobija se rezultat prikazan na slici 17.

```
PRIMARY> rs.remove("mongod2.replicaset.com:27017")
{ "ok" : 1 }
```

Slika 17

Za potvrdu da li je član uspešno uklonjen, izdaje se `rs.status()` naredba.

4.4.5. Dodavanje arbitra

Čvor arbitra ne čuva nikakve podatke, već samo glasa u izbronom procesu, i obično se dodaje skupu replika sa parnim brojem članova. S obzirom da ne skladišti podatke, da bi se poboljšale performanse arbitra, onemogućuju se neke funkcije. To se može učiniti sledećim koracima:

1. Za pokretanje arbitra potrebno je kreirati dbpath direktorijum koji se koristi za čuvanje konfiguracija, ne podataka. Direktorijum se kreira pomoću sledeće naredbe komandne linije:
`mkdir /data/arbiter`
2. Zatim se mongod instanca pokreće koristeći kreirani dbpath direktorijum:
`mongod --port 20000 --dbpath /data/arbiter --replSet rs1`
3. Nakon pokretanja arbitražnog čvora, on se dodaje skupu replika pomoću naredbe:
`rs.addArb("arbiter1.replicaset.com:20000")`, nakon čega se dobija rezultat prikazan na slici 18.

```
rs1:PRIMARY> rs.addArb("arbiter1.replicaset.com:20000")
{ "ok" : 1 }
```

Slika 18

4.4.6. Testiranje metoda zaštite sistema od kvara²

Da bi se replika prisilila da izvrši izborni proces može se deaktivirati određeni čvor.

Prvo deaktiviramo jedan od sekundarnih čvorova zaustavljanjem mongod servisa na sekundarnoj mašini, mongod3.replicaset.com, nakon čega će ovaj čvor biti nedostupan, što je prikazano na slici 19. Nakon ponovnog pokretanja servisa, skup replika dobija ping od trećeg člana, i član ponovo postaje dostupan, što je prikazano na slici 20.

```
rs1:PRIMARY> rs.status()
{
  "set" : "rs1",
  "date" : ISODate("2011-06-11T09:34:40Z"),
  "members" : 3,
  "syncingTo" : 0,
  "optime" : {
    "t" : 1397231800,
    "i" : 1
  },
  "primary" : "mongod1.replicaset.com:27017",
  "secondary" : [
    {
      "_id" : 2,
      "name" : "mongod3.replicaset.com:27017",
      "health" : 1,
      "state" : 2,
      "stateStr" : "SECONDARY",
      "uptime" : 1,
      "optime" : {
        "t" : 1397231800,
        "i" : 1
      },
      "syncingTo" : "mongod1.replicaset.com:27017",
      "error" : "socket exception"
    },
    {
      "_id" : 3,
      "name" : "mongod2.replicaset.com:27017",
      "health" : 1,
      "state" : 1,
      "stateStr" : "PRIMARY",
      "uptime" : 1,
      "optime" : {
        "t" : 1397231800,
        "i" : 1
      },
      "syncingTo" : null
    }
  ]
}
```

Slika 19

```
{
  "_id" : 2,
  "name" : "mongod3.replicaset.com:27017",
  "health" : 1,
  "state" : 2,
  "stateStr" : "SECONDARY",
  "uptime" : 1,
  "optime" : {
    "t" : 1397231918000,
    "i" : 1
  },
}
```

Slika 20

² Engl. *failover* – metod zaštite sistema od kvara, pri čemu oprema u stanju pripravnosti automatski preuzima rad u slučaju otkaza glavnog sistema

Za sledeći test zaustavljamo servis primarnog čvora. Nakon zaustavljanja servisa u primarnom čvoru, skup replika pokreće izborni postupak da bi izabrao novi primarni čvor od sekundarnih čvorova.

Izlaz `rs.status()` naredbe nakon zaustavljanja primarnog čvora prikazano je na slici 21. Vidimo da će drugi sekundarni čvorovi sinhronizovati svoje podatke sa novim primarnim čvorom, koji je u ovom primeru `mongod3.replicaset.com`

```
{
  "_id" : 1,
  "name" : "mongod2.replicaset.com:27017",
  "health" : 1,
  "state" : 2,
  "stateStr" : "SECONDARY",
  "uptime" : 290,
  "optime" : Timestamp(1397231918, 1),
  "optimeDate" : ISODate("2021-08-11T15:58:38Z"),
  "lastHeartbeat" : ISODate("2021-08-11T20:46:28Z"),
  "lastHeartbeatRecv" : ISODate("2021-08-11T20:46:29Z"),
  "pingMs" : 4,
  "lastHeartbeatMessage" : "syncing to: mongod3.replicaset.com:27017",
  "syncingTo" : "mongod3.replicaset.com:27017"
},
{
  "_id" : 2,
  "name" : "mongod3.replicaset.com:27017",
  "health" : 1,
  "state" : 1,
  "stateStr" : "PRIMARY",
  "uptime" : 290,
  "optime" : Timestamp(1397231918, 1),
  "optimeDate" : ISODate("2021-08-11T15:58:38Z"),
  "self" : true
},
{
  "ok" : 1
}
```

Slika 21

Status prethodnog primarnog čvora, `mongod1.replicaset.com`, prikazan je na slici 22.

```
{
  "_id" : 0,
  "name" : "mongod1.replicaset.com:27017",
  "health" : 0,
  "state" : 8,
  "stateStr" : "(not reachable/healthy)",
  "uptime" : 0,
  "optime" : Timestamp(1397231918, 1),
  "optimeDate" : ISODate("2021-08-11T15:58:38Z"),
  "lastHeartbeat" : ISODate("2021-08-11T20:46:28Z"),
  "lastHeartbeatRecv" : ISODate("2021-08-11T20:45:14Z"),
  "pingMs" : 0
},
```

Slika 22

Da bismo završili naš scenario testiranja, ponovo pokrećemo servis sa `mongod1.replicaset.com` mašine. Odmah nakon omogućavanja usluge, `mongod` će poslati ping poruke trenutnom primarnom čvoru, i `mongod1.replicaset.com` postaje novi, sada sekundarni čvor. Na slici 23 prikazan je rezultat ponovnog pokretanja `mongod` servisa.

```

"members" : [
  {
    "_id" : 0,
    "name" : "mongod1.replicaset.com:27017",
    "health" : 1,
    "state" : 2,
    "stateStr" : "SECONDARY",
    "uptime" : 2,
    "optime" : Timestamp(1397231918, 1),
    "optimeDate" : ISODate("2021-08-11T15:58:38Z"),
    "lastHeartbeat" : ISODate("2021-08-11T20:55:55Z"),
    "lastHeartbeatRecv" : ISODate("2021-08-11T20:45:14Z"),
    "pingMs" : 0
  },
  {
    "_id" : 1,
    "name" : "mongod2.replicaset.com:27017",
    "health" : 1,
    "state" : 2,
    "stateStr" : "SECONDARY",
    "uptime" : 858,
    "optime" : Timestamp(1397231918, 1),
    "optimeDate" : ISODate("2021-08-11T15:58:38Z"),
    "lastHeartbeat" : ISODate("2021-08-11T20:55:56Z"),
    "lastHeartbeatRecv" : ISODate("2021-08-11T20:55:55Z"),
    "pingMs" : 0,
    "syncingTo" : "mongod3.replicaset.com:27017"
  },

```

Slika 23

5. Koncept deljenja baze podataka

U ovom poglavlju biće reči o jednoj od najcenjenijih karakteristika MongoDB baze podataka, odnosno o deljenju. Korišćenjem funkcije deljenja podaci se distribuiraju na različite servere kako bi se distribuiralo i opterećenje i poboljšale performanse. Nakon upoznavanja sa konceptima deljenja, u drugom delu poglavlja, konfiguriše se server i podešava od nule.

5.1. Skaliranje

Skladištenje velike količine podataka na jednu mašinu može suočiti MongoDB bazu podataka sa slabom propusnošću ili sa nedostatkom serverskih resursa. Da bi se rešili ovi problemi, sistem baze podataka treba da se prilagoditi tako da podržava više klijenata.

U osnovi postoje dva načina za skaliranje sistema, vertikalno i horizontalno.

Kod vertikalnog skaliranja, sistemski administrator dodaje više resursa ili povećava kapacitet postojećih resursa serveru. S druge strane, metoda horizontalnog skaliranja povećava kapacitet čitavog servera spajanjem manjih instanci. O vertikalnom i horizontalnom skaliranju već je bilo reči u poglavlju 3.2.

5.2. Deljenje

Deljenje (engl. *sharding*) je tehnika koju MongoDB baza podataka koristi za horizontalno skaliranje. Korišćenjem deljenja, cela baza podataka se deli u zasebne pojedinačne baze podataka na različitim mašinama. Pomoću usmerivača upita (engl. *query router*) odvojeni čvorovi čine jednu logičku bazu podataka.

Primer baze podataka od 1TB koja je podeljena na četiri instance razmatran je u poglavlju 3.2 i prikazan na slici 5.

Prednosti korišćenja deljenja nisu ograničene samo na smanjenje veličine baze podataka podelom na manje instance. Uz pomoć deljenja može se podržati veći broj zahteva klijenata i poboljšati performanse. Klijent pristupa samo jednom delu, a ne celoj bazi podataka, pa baza može podržati više zahteva.

5.2.1. Moduli podele

Deljenja baze podataka, MongoDB baza implementira pomoću različitih modula i komponenti, pri čemu je svaki moduo odgovoran za određeni zadatak.

Prvi moduo je deo (engl. *shard*) koji je odgovoran za čuvanje podataka.

Sledeća komponenta je ruter za upite. Zadatak rutera za upite je već objašnjen u poglavlju 3.3, samo se zahtevi sada, umesto do skupa replika, usmeravaju do odgovarajućeg dela. Izdeljeni klaster može da sadrži jedan ili više rutera za upite.

Poslednji moduo je konfiguracioni server, koji prikuplja metapodatke o izdeljenim klasterima, konfiguracijama delova, statusu svakog čvora i odnosu između njih. U proizvodnom okruženju, izdeljeni klaster ima tri konfiguraciona servera.

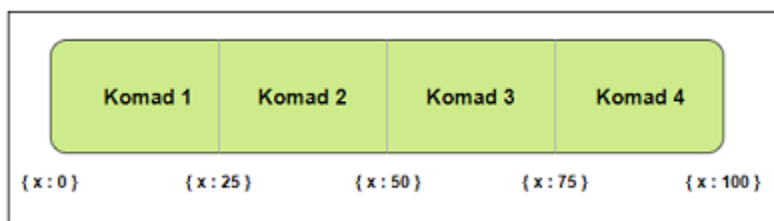
Izdeljeni klaster i odnos između komponenti prikazan je na slici 6.

5.2.2. Korišćenje ključeva za deljenje

Izdeljeni klaster koristi ključ za deljenje da bi se ceo skup podataka podelio na manje delove. Ključ za deljenje je indeksirani ključ ili indeksirani složeni ključ koji postoji u svim zapisima u kolekciji. Nakon definisanja ključa za deljenje i veličine svakog dela, skup podataka se deli na manje delove i delovi se sinhronizuju.

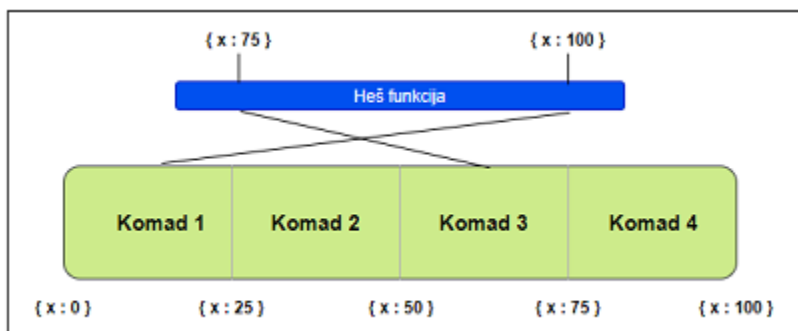
Ključ za deljenje može biti zasnovan na opsegu ili na hešu.

Koristeći ključeve zasnovane na opegu, mehanizam baze podataka deli skup podataka na manje delove sa određenim ospezima vrednosti datog polja. Slika 24 ilustruje podelu zasnovanu na opsegu, koje je izvršena na osnovu polja koje ima numeričku vrednost koja se kreće od 0 do 100, pri čemu je baza podeljena na četiri dela (engl. *chunks*).



Slika 24: Podela zasnovana na opsegu

U metodi koja koristi ključ zasnovan na heširanju, pomoću heš funkcije se hešira vrednost određenog polja. Kada klijent zahteva podatke, MongoDB koristi heš funkciju da pronade lokaciju svakog heš ključa u određenom delu baze podataka (komadu), a zatim usmerava klijenta na određeni deo. Ova tehnika osigurava da se kolekcije nasumično dele između delova. Izdeljeni klaster koji koristi ključeve zasnovane na hešu prikazan je na slici 25.



Slika 25: Podela koja koristi ključ zasnovan na heširanju

5.3. Proces cepanja i uravnoteženja

Cepanje i uravnoteženje su dva pozadinska procesa koja služe za održavanje čitavog klastera. Proces cepanja sprečava da se delovi prekomerno uvećaju i kontroliše veličinu svakog dela. Kada veličina dela postane veća od definisanog praga, taj deo se deli na pola.

S druge strane, zadatak procesa balansiranja je migriranje delova sa jednog dela (engl. *shard*) na drugi. Kada broj delova u komadima nije ujednačen, proces balansiranja migrira komadić iz dela koji ima veći broj komada u onaj koji ima manje. MongoDB migrira komad sa izvorišnog na određeni deo, a tek kada je migracija uspešno završena, uklanja komadić sa izvornog dela.

5.4. Praktičan primer deljenja baze podataka

Da bi se omogućilo konfigurisanje i podešavanje mreže klastera, prvo moraju da se konfiguriraju konfiguracioni serveri koji sadrže metapodatke mreže klastera i informacije o delovima (engl. *shard*). Ostali delovi mreže klastera koriste ove konfiguracione servere da bi dobili podatke o drugim delovima.

U proizvodnom okruženju se preporučuje imati tačno tri konfiguraciona servera na različitim mašinama. Razlog postavljanja svakog dela na posebnom serveru je poboljšanje sigurnosti podataka i čvorova.

Svaki konfiguracioni server ima adresu koja usmerava do ciljne mašine. U ovom primeru imamo tačno tri konfiguraciona servera, a sledeća lista prikazuje imena hostova:

- cfg1.sharding.com
- cfg2.sharding.com
- cfg3.sharding.com

U našem primeru ćemo postaviti demo funkciju deljenja, pa raspoređujemo sve konfiguracione servere na jednu mašinu sa različitim portovima. Za proizvodnu upotrebu, sve će biti isto, osim što je potrebno hostovati konfiguracione servere na različite mašine.

U nastavku su implementirani delovi konfiguracionog servera i sve povezano kako bi se pokrenuo server za deljenje baze podataka i mreža klastera.

5.4.1. Implementacija konfiguracionih servera

Uspostavljanje konfiguracionog servera je zapravo pokretanje mongod instance pomoću parametra `--configsvr`. Struktura naredbe je:

```
mongod --configsvr --dbpath <path> --port <port>
```

Ako se ne proslede parametri `dbpath` ili `port`, konfiguracioni server koristi `/data/configdb` kao putanju za skladištenje podataka i port 27019 za izvršavanje instance. Međutim, podrazumevane vrednosti se mogu promeniti pomoću prethodne naredbe.

Pre pokretanja konfiguracionog servera uverite se da ste kreirali `dbpath` putanju, da ne biste dobili grešku. Takođe treba voditi računa da se mongod instanca izvršava sa dovoljnim nivoom dozvole, root ili administratorskim nivoom dozvole.

Nakon izvršavanja naredbe trebalo bi dobiti rezultat prikazan na slici 26. Vidimo da imamo konfiguracioni server sa imenom hosta `cfg1.sharding.com`, sa portom 27019 i sa `/data/configdb` kao `dbpath`.

Takođe, postoji veb konzola za upravljanje konfiguracionim serverom koji radi na portu 28019. Veb konzolu možete videti usmeravanjem pretraživača na adresu <http://localhost:28019/>.

Sada imamo prvi konfiguracioni server koji radi. Na isti način se pokreću druge instance, to jest koristeći `/data/configdb2` sa portom 27020 za drugi, i `/data/configdb3` sa portom 27021 za treći konfiguracioni server.

5.4.2. Konfigurisanje rutera za upite (mongos instance)

Nakon konfigurisanja konfiguracionih servera, treba ih povezati sa osnovnim modulom klasterisanja. Instanca mongos procesa odgovorna je za povezivanje svih modula i delova. Proces mongos podrazumevano koristi port 27017, ali se on može promeniti pomoću konfiguracionih parametara.


Za definisanje konfiguracionih servera koristi se konfiguraciona datoteka ili parametri komandne linije. Napravite novu `mongos.conf` datoteku u `/etc/` direktorijumu i u datoteku upišite podešavanja za konfigurisanje:

`configdb = cfg1.sharding.com:27019, cfg2.sharding.com:27020, cfg3.sharding.com:27021`

Za pokretanje mongos instance koristi se sledeća komanda:

`mongos -f /etc/mongos.conf`

Dobijeni rezultat je prikazan na slici 26.



```
root@mongo:~# sudo mongos --config
[initandlisten] MongoDB starting : pid=3984 port=27017 dbpath=/data/configdb 64-bit host=cfg1.sharding.com
[initandlisten] db version v2.6.0, pointTo version 4.5
[initandlisten] git version: 005d70ff9f9c301a850a0130a0f0270a446222
[initandlisten] build info: Darwin armv7, libmcc 9.6.0 Darwin kernel version 9.6.0: Mon Nov 24 17:37:09 PST 2008;
root@mongo:~# mongos --config
[initandlisten] options: { configdb: true }
[initandlisten] journal dir=/data/configdb/journal
[initandlisten] recover: no journal files present, no recovery needed
[initandlisten] admin web console waiting for connections on port 28019
[initandlisten] waiting for connections on port 27017
```

Slika 26

Nakon ovog dela imamo mongos i sve konfiguracione servere. Treba da dodamo delove na mongos instancu da bi mreža bila kompletirana.

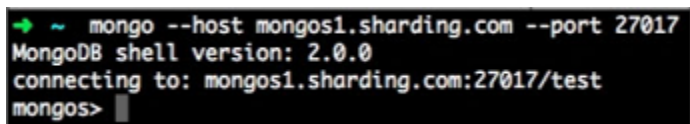
5.4.3. Upravljanje mongos instancom

Sada dodajemo delove (engl. *shards*) i delimo ceo skup podataka na manje delove.

Da bismo kontrolisali i upravljali mongos instancom, koristimo mongo ljsku za povezivanje sa mongos instancama i izvršenje naredbi. Da biste se povezali sa mongos instancom, koristi se sledeća komanda:

`mongo --host <mongos hostname> --port <mongos port>`

Na slici 27 se vidi da je naša mongos adresa `mongos1.sharding.com`, a port je 27017.



```
➔ ~ mongo --host mongos1.sharding.com --port 27017
MongoDB shell version: 2.0.0
connecting to: mongos1.sharding.com:27017/test
mongos>
```

Slika 27

Nakon povezivanja sa mongos instancom, imamo komandno okruženje koje se može koristiti za dodavanje, uklanjanje ili modifikovanje delova ili za dobijanje statusa cele mreže za deljenje.

5.4.4. Dodavanje delova mongos procesu

MongoDB ima prostor imena (engl. *namespace*) `sh` i u njemu funkciju `addShard()` koja se koristi za dodavanje novog dela postojećoj mreži za deljenje. Na slici 28 je prikazan poziv ove funkcije.



```
sh.addShard( host ) server:port OR setname/server:port
```

Slika 28

Za dodavanje skupa replika mongos procesu, treba pratiti ovu šemu: `setname/server:port`

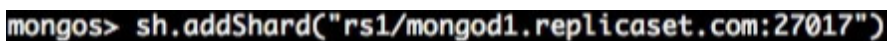
Na primer, ako imamo skup replika sa imenom `rs1`, imenom hosta `mongod1.replicaset.com` i brojem porta 27017, naredba za dodavanje skupa replika mongos procesu će biti sledeća:

`sh.addShard("rs1/mongod1.replicaset.com:27017")`

Koristeći istu funkciju možemo dodati i samostalne mongod instance. Na primer, ako imamo mongod instance sa imenom hosta `mongod1.sharding.com` koja sluša na portu 27017, naredba će biti sledeća:

`sh.addShard("mongod1.sharding.com:27017")`

U našem primeru dodajemo mrežu skupova replika koju smo napravili u prethodnom poglavlju. Ako sve prođe kako treba, ne prikazuju se nikakvi rezultati na konzoli, što znači da je postupak dodavanja bio uspešan. Slika 29 prikazuje izvršenje naredbe.



```
mongos> sh.addShard("rs1/mongod1.replicaset.com:27017")
```

Slika 29

Na slici 30 prikazan je status deljenja, koji je dobijen izvršavanjem `sh.status()` komande.

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: { "_id" : 1, "version" : 3 }
    shards:
      { "_id" : "rs1", "host" : "rs1/mongod1.replicaset.com:27017,mongod3.replicaset.com:27017,mongod2.replicaset.com:27017" }
    databases:
      { "_id" : "admin", "partitioned" : false, "primary" : "config" }
      { "_id" : "test", "partitioned" : false, "primary" : "rs1" }
mongos>
```

Slika 30

Slika 30 prikazuje pokretanje još jedne mongod instance koja će biti uključena u proces deljenja. Broj porta mongod procesa je 27016, a ime hosta mongod1.sharding.com. Dodavanje ovog čvora u proces deljenja je prikazano na slici 31.

```
mongos> sh.addShard("mongod1.sharding.com:27016")
mongos>
```

Slika 31

Status mreže za deljenje nakon dodavanja čvora prikazan je na slici 32. Na slici se vidi da sada imamo dva dela. Prvi je skup replika sa imenom rs1, a drugi je samostalna mongod instance na portu 27016.

Sada imamo sve module za deljenje koji rade zajedno. Poslednji korak je omogućavanje deljenja baze podataka i kolekcije.

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: { "_id" : 1, "version" : 3 }
    shards:
      { "_id" : "rs1", "host" : "rs1/mongod1.replicaset.com:27017,mongod3.replicaset.com:27017,mongod2.replicaset.com:27017" }
      { "_id" : "shard0000", "host" : "mongod1.sharding.com:27016" }
    databases:
      { "_id" : "admin", "partitioned" : false, "primary" : "config" }
      { "_id" : "test", "partitioned" : false, "primary" : "rs1" }
mongos>
```

Slika 32

5.4.5. Omogućavanje deljenja

Da bi se omogućilo deljenje baze podataka koristi se mongo okruženje komandne linije. Deljenje baze podataka se omogućava korišćenjem sledeće naredbe:

`sh.enableSharding(database)`

Slika 33 prikazuje omogućavanje deljenja newdb baze podataka. Sa slike vidimo da je polje partitioned tačno za newdb bazu podataka, a netačno za ostale baze.

```

mongos> sh.status()
--- Sharding Status ---
  sharding version: { "_id" : 1, "version" : 3 }
  shards:
    [ { "_id" : "rs1", "host" : "rs1/mongod1.replicaset.com:27017,mongod3.replicaset.com:27017,mongod2.replicaset.com:27017" }
      { "_id" : "shard0000", "host" : "mongod1.sharding.com:27016" }
    ]
  databases:
    [ { "_id" : "admin", "partitioned" : false, "primary" : "config" }
      { "_id" : "test", "partitioned" : false, "primary" : "rs1" }
      { "_id" : "newdb", "partitioned" : true, "primary" : "shard0000" }
    ]
mongos>

```

Slika 33

Pretpostavimo da baza podataka newdb ima probnu kolekciju sa sledećom šemom:

```

{
  "_id": ObjectId("725c211a412f812548cv3258"),
  "data":1
}

```

Ključ za deljenje će biti `_id`. MongoDB će distribuirati podatke pomoću `_id` polja. Potrebno je kreirati indeks na kolekciji za ključ za deljenje pomoću sledeće komande:

```
db.test.ensureIndex( { _id : "hashed" } )
```

Rezultat izvršenja ove naredbe prikazan je na slici 34.

```

mongos> db.test.ensureIndex( { _id : "hashed" } )
{
  "raw" : {
    "mongod1.sharding.com:27016" : {
      "note" : "downgraded",
      "sentTo" : "mongod1.sharding.com:27016",
      "eachIndex" : [
        {
          "spec" : {
            "key" : {
              "_id" : "hashed"
            },
            "name" : "_id_hashed",
            "ns" : "newdb.test"
          },
          "gle" : {
            "n" : 0,
            "connectionId" : 15,
            "err" : null,
            "ok" : 1
          }
        }
      ]
    },
    "ok" : 1
  },
  "ok" : 1
}
mongos>

```

Slika 34

Deljenje kolekcije možemo omogućiti idavanjem dole navedene naredbe. Rezultat izvršenja ove naredbe prikazan je na slici 35.

```
sh.shardCollection("newdb.test", { "_id": "hashed" } )
```

```
mongos> sh.shardCollection("newdb.test", { "_id": "hashed" } )
{ "collectionsharded" : "newdb.test", "ok" : 1 }
mongos> █
```

Slika 35

Nakon omogućavanja deljenja, naš status deljenja prikazan je na slici 36.

```
shards:
  { "_id": "rs1", "host": "rs1/mongod1.replicaset.com:27017,mongod2.replicaset.com:27017,mongod3.replicaset.com:27017" }
  { "_id": "shard0000", "host": "mongod1.sharding.com:27016" }
databases:
  { "_id": "admin", "partitioned": false, "primary": "config" }
  { "_id": "newdb", "partitioned": true, "primary": "shard0000" }
newdb.test
  shard key: { "_id": "hashed" }
  chunks:
    rs1      2
    shard0000 2
    { "_id": { "$key": 1 } } --> { "_id": NumberLong("-4611686018427387902") } on : rs1 Timestamp(2, 2)
    { "_id": NumberLong("-4611686018427387902") } --> { "_id": NumberLong(0) } on : rs1 Timestamp(2, 3)
    { "_id": NumberLong(0) } --> { "_id": NumberLong("4611686018427387902") } on : shard0000 Timestamp(2, 4)
    { "_id": NumberLong("4611686018427387902") } --> { "_id": { "$key": 1 } } on : shard0000 Timestamp(2, 5)
```

Slika 36

Da bismo testirali funkciju deljenja, ubacujemo podatke u kolekciju u newdb bazu podataka. Mongo ljsuka je JavaScript okruženje i u njoj je moguće pokretati JavaScript kod. Pomoću JavaScript koda ubacujemo 10.000 podataka u kolekciju, što je prikazano na slici 37.

```
mongos> for (var i = 1; i <= 10000; i++) db.test.insert( { data : i } )
WriteResult({ "nInserted" : 1 })
```

Slika 37

Pošto trenutno imamo dva dela, MongoDB deli podatke na dva komada. Ako pokrenemo naredbu `db.test.find().count()` da bismo dobili broj podataka u skupu replika, videćemo manje brojeva od onoga što smo ubacili pomoću JavaScript koda (10.000 brojeva). Izlaz iz prethodne naredbe prikazan je na slici 38. Sa slike vidimo da u skupu replika imamo 5.073 brojeva. Ako izvršimo isti upit nad samostalnom mongod instancom, vidimo da je ostatak podataka u ovom delu, kao što je prikazano na slici 39.

```
rs1:PRIMARY> db.test.find().count()
5073
rs1:PRIMARY> █
```

Slika 38

```
> show dbs
admin (empty)
local (empty)
newdb 0.203125GB
> use newdb
switched to db newdb
> db.test.find().count()
4927
> █
```

Slika 39

6. Analiza i poboljšanje performansi baze podataka

U ovom poglavlju ćemo izučiti načine i metode za poboljšanje vremena odziva i performansi operacija čitanja/pisanja.

6.1. Profilisanje

Za suočavanje sa lošim performansama aplikacije, potreban je alat za pronalaženje i rešavanje problema. Profilisanje³ je popularan metod za otklanjanje grešaka i pronalaženje delova koji pogoršavaju performance programa.

6.1.1. Korišćenje profilisanja

Pisanje optimizovanih upita jedan je od najvažnijih faktora za poboljšanje performansi baze podataka. Profilisanje baze podataka i upita metoda je koja se koristi za pronalaženje uskih grla i rešavanje problema sa performansama.

MongoDB podržava profilisanje za svaku instancu ili za svaku bazu podataka. Korišćenjem profilisanja mogu se pronaći najsporiji upiti u programu i rešiti problemi zamenom starih upita optimizovanim verzijama.

MongoDB ima mehanizam za profilisanje koji prikuplja podatke u `system.profile` kolekciji.

Mašina za profilisanje podržava različite nivoe profilisanja. Sledeća lista prikazuje različite nivoe profilisanja dostupne u MongoDB bazi podataka:

- **Nivo 0:** Mašina za profilisanje je isključena i na ovom nivou mašina za profilisanje ne evidentira ništa. Ovo je podrazumevani nivo profilisanja.
- **Nivo 1:** Mašina za profilisanje prikuplja podatke pod određenim uslovima, na primer samo za operacije koju su sporije od određenog praga.

³ U softverskom inženjerstvu, profilisanje je oblik dinamičke analize programa koji meri prostor (memoriju) ili vreme, složenost programa ili učestalost i trajanje poziva funkcije. Profilisanje pomaže u optimizaciji programa. [6]

- **Nivo 2:** Mehanizam za profilisanje prikuplja podatke o svim operacijama u bazi podataka.

Mašina za profilisanje koristi `operationProfiling.slowOpThresholdMs` opciju da bi operaciju smatrala sporom operacijom. Podrazumevana vrednost za ovu opciju je 100ms, ali se može jednostavno promeniti konfigurisanjem opcije drugom vrednošću u `mongod` konfiguracionoj datoteci.

6.1.2. Omogućavanje i konfigurisanje profila

Omogućavanje, konfigurisanje i korišćenje mehanizama za profilisanje vrši se preko mongo ljske.

Struktura funkcije `setProfilingLevel` je sledeća:

`db.setProfilingLevel(level, slowms)`

Prvi parametar je nivo profila, koji je obavezan parametar. Drugi opciona parametar, `slowms`, je prag u milisekundama da bi se operacija smatrala sporom operacijom.

Za omogućavanje profilera za bazu podataka, koristi se sledeća komanda u interfejsu mongo ljske:

`db.setProfilingLevel(2)`

Prethodna naredba omogućava profilisanje i postavlja opširnost profila na najviši nivo. Profiler će evidentirati sve operacije i sačuvati ih u `system.profile` kolekciji.

Sledećom naredbom se u isto vreme podešava nivo profilisanja na 1 i konfiguriše opcija `slowOpThresholdMs` na 50ms:

`db.setProfilingLevel(1, 50)`

Da bi se dobio trenutni status profila, može se koristiti sledeća komanda u mongo ljski:

`db.getProfilingStatus()`

Izlaz iz prethodne naredbe prikazan je na slici 40. U polju `was` se vidi trenutni nivo profila, a vrednost `slowms` argumenta `setProfilingLevel` funkcije se vidi u `slowOpThresholdMs` polju.


```
> db.getProfilingStatus()
{ "was" : 1, "slowms" : 10 }
> █
```

Slika 40

6.2. Druge analitičke metode

MongoDB nudi `explain()` funkciju, koja pruža detaljan postupak za trenutni operator. Razmotrimo sledeću naredbu:

```
db.testcollection.find().sort({x: -1}).explain()
```

Ova naredba izvršava operator pronalaženja i sortira zapise u silaznom poretku na osnovu polja `x`. Izlaz funkcije sadrži, između ostalog, broj pogođenih zapisa, vreme provedeno za izvršavanje operatora u milisekundama, logičku vrednost o tome da li su vraćena polja već indeksirana... Izlaz koji vraća funkcija se prijazuje u JSON formatu.

Dalje, MongoDB pruža mogućnost programerima da dobiju izveštaj o tekućim operacijama pomoću funkcije `db.currentOp()`. Ova funkcija prihvata jedan opcioni logički parametar koji određuje režim detaljnosti funkcije.

6.3. Uvođenje indeksa

Radi poboljšanja performansi baze podataka mogu se koristiti indeksi za kategorizaciju zapisa. Uvođenjem indeksa upiti za čitanje/pisanje se mogu izvršiti brže nego ranije.

Ako se često radi sa nekim poljima iz kolekcije, preporučuje se definisanje indeksa za njih kako bi se poboljšale performance. Na primer, ako se često vrši sortiranje na osnovu određenog polja, preporučljivo je napraviti indeks nad tim poljem kako bi mehanizam baze podataka mogao brže da izvrši upit.

Za definisanje indeksa za određeno polje, na primer za polje naslova kolekcije vesti, može se koristiti sledeća naredba:

```
db.news.ensureIndex({ title: 1 })
```

Nakon kreiranja indeksa, ako se izda sledeća naredba, rezultat će se dobiti brže nego ranije:

```
db.news.find().sort({ title: -1 })
```

6.4. Korišćenje projekcije

Još jedan pristup za brže izvršavanje upita u MongoDB bazi podataka je korišćenje funkcije projekcije. Projekcijom se ograničava izlaz baze podataka na određena polja i suvišni podaci se uklanjaju iz rezultata. Rezultat je zahvaljujući tome manji i klijent dobija rezultat brže nego ranije.

Sledi primer naredbe koje vraća zapise koje za polje x imaju vrednost veću od 500 i gde je polje _id isključeno iz rezultata:

```
db.testcollection.find({ x: { $gt: 500 } }, { x: 1, _id: 0 })
```

Treba napomenuti da prenošenje vrednosti 0 na opciju projekcije isključuje polje iz konačnog rezultata, a 1 ga uključuje.

6.5. Ograničavanje broja vraćenih zapisa

Još jedan trik za poboljšanje performansi operacija je ograničavanje broja zapisa. Naime, mehanizam vraća samo podskup svih zapisa i klijent preuzima manji deo podataka.

Da bi se ograničio broj vraćenih stavki koristi se funkcija limit(), kao što je prikazano ispod:

```
db.testcollection.find().limit(2)
```

6.6. Poboljšanja na nivou hardvera

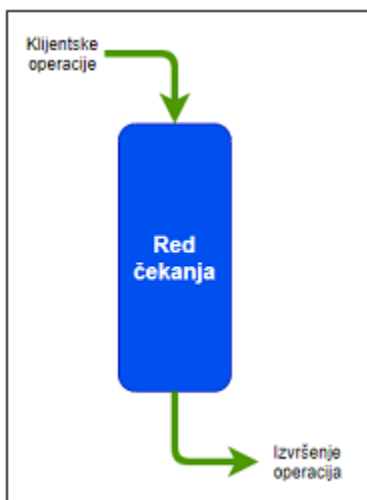
Promena hardvera mašine radi poboljšanja performansi baze podataka je uobičajen obrazac među administratorima baze podataka. Na primer, korišćenje SSD diska može poboljšati performance čitanja/pisanja. Takođe, za mongod procese je neophodno da mašina ima dovoljno RAM memorije. Ako nema dovoljno RAM-a na jednom serveru, koriste se klaster rešenja i RAM kapaciteti svih mašina uključenih u klaster.

7. Migracija instanci i smanjenje zastoja

Migracija MongoDB instance na drugu mašinu je česta situacija sa kojom se administratori suočavaju.

Uobičajeni metod premeštanja instance je instaliranje mehanizma baze podataka na novi server, kopiranje putanje baze podataka sa postojećeg servera na novi server, a zatim pokretanje nove instance.

Međutim, u toku migracije može doći do zastoja jer klijenti ne mogu obavljati operacije pisanja. MongoDB uvodi red čekanja i stavlja operacije pisanja u red ako su suočene sa greškom, i pokušava da ih upiše u bazu u određenim vremenskim intervalima. Red čekanja i odnos između klijenata i MongoDB mašine prikazan je na slici 41.



Slika 41: Red čekanja

Pomoću funkcije `slaveOk(true)` omogućava se da se operacije čitanja usmeravaju na sekundarne članove kako bi baza podataka tokom migracije funkcionisala bez zastoja.

Detalji procesa migracije instanci i rešenja problema zastoja u radu koji nastaje prilikom migracije instanci prevazilaze obim ovog rada.

8. Nadgledanje baze podataka i rešavanje problema nad bazom podataka

Nadgledanje je jedan od kritičnih zadataka administracije baze podataka. Korišćenje ispravnih alata za nadzor mogu pomoći administratorima baza podataka da obezbede lako dostupan mehanizam baze podataka.

MongoDB ima mnogo alata za praćenje i profilisanje. Za dobijanje izveštaja baze podataka u realnom vremenu ili za proveru trenutno pokrenutih operacija čitanja/pisanja, mogu se koristiti ugrađene konande ili uslužni programi. S druge strane, za dijagnostifikovanje baze podataka mogu se koristiti alati za profilisanje.

Takođe, postoji veliki broj alata za nadzor zasnovanih na internetu.

Različite kategorije praćenja i dijagnostifikovanja baze podataka prikazane su na slici 42.



Slika 42

U nastavku ovog rada biće objašnjena svaka gorepomenuta kategorija.

8.1. Korišćenje funkcije profilisanja

Profilisanje je tehnika za otkrivanje sporih operacija. Pomoću nje se mogu pronaći neefikasne operacije i poboljšati performance menjanjem operacija i upita. Funkcija profilisanja je detaljnije objašnjena u poglavlju 6, Analiza i poboljšanje performansi baze podataka.

8.2. Korišćenje ugrađenih alata za izveštavanje

MongoDB ima neke ugrađene alate za izveštavanje.

Ugrađeni alati za izveštavanje su:

- **mongotop** – uslužni program za dobijanje statistike MongoDB baze podataka u realnom vremenu. Izveštava o trenutnim operacijama čitanja/pisanja dostupne instance. Mongotop prikazuje izveštaj po bazi podatka i kolekciji. Za korišćenje ovog uslužnog programa potrebno je izdati mongotop naredbu u interfejsu komandne linije.
- **mongostat** – uslužni program koji pruža izveštaj o performansama MongoDB baze podataka u realnom vremenu. Prikazuje broj operacija po bazi podataka. Za korišćenje ovog programa potrebno je izdati mongostat naredbu u interfejsu komandne linije.

8.3. Korišćenje uslužnih programa zasnovanih na internetu

Alati iz ove kategorije su uglavnom jednostavni za upotrebu i konfigurisanje.

Neki od uslužnih programa zasnovanih na internetu su:

- **MMS** – olakšava upravljanje MongoDB instancom. Podržava sve vrste MongoDB instance, uključujući samostalne instance, izdvojene klastere, master/slave i skup replika.
- **Scout** – izviđač je alat za nadzor servera poslovnog nivoa. Pruža različite dodatke za upravljanje skupovima replika i konfigurisanje poruka upozorenja. Scout program nema besplatne verzije, ali nude besplatan probni period.
- **Server density** – alat koji pruža moduo za upravljanje MongoDB instancama. Pruža funkcije koje programerima i administratorima mogu pomoći da bez napora upravljaju bazom podataka. Ovaj alat takođe nema besplatne verzije, ali je moguće dobiti besplatnu probnu verziju.
- **LogicMonitor** – alat za nadzor koji podržava različite vrste sistema uzbunjivanja. Ne postoji besplatna verzija ovog alata.
- **FusionReactor** – alat za upravljanje u realnom vremenu. Ni za ovaj alat ne postoje besplatne verzije.

9. Zaključak

Visoka dostupnost (engl. *high availability*) je sposobnost sistema da radi neprekidno, bez greške u određenom vremenskom periodu.

Koncept visoke dostupnosti se odnosi i na podatke, jer bez podataka aplikacije i usluge ne mogu da funkcionišu. U okviru ovog rada teorijski je obrađena tema visoke dostupnosti baza podataka, a praktičnim primerima je pokriveno funkcionisanje skupa replika i deljenja baze podataka.

Najčešći razlozi neuspeha MongoDB baze podataka su zastoj servera, greška pri upisu ili čitanju i tako dalje. Korišćenje 32-bitne verzije MongoDB baze podataka umesto 64-bitne, može takođe stvoriti određene probleme, pa se preporučuje isključivo korišćenje 64-bitne verzije MongoDB baze podataka.

Skup replika je kolekcija procesa, od kojih je samo jedan primarni proces, a ostali su sekundarni. Članovi skupa replika mogu biti primarni, sekundarni i abitrarni čvorovi. Replikacija je postupak sinhronizacije podataka između primarnog i sekundarnih čvorova.

Sistemi baza podataka se mogu skalirati na dva načina, vertikalno i horizontalno. Kod vertikalnog skaliranja potrebno je dodati veći kapacitet postojećem serveru, dok se kod horizontalnog skaliranja, sistem baze podataka širi kroz različite servere.

MongoDB baze podataka podržavaju koncept deljenja, kojim se podaci distribuiraju na različite servere kako bi se distribuiralo i opterećenje i poboljšale performanse. Deljenje baze se vrši pomoću ključa za deljenje, koji može biti zasnovan na opsegu ili hešu.

Podeljeni klaster (engl. *sharded cluster*) u MongoDB bazi podataka se sastoji od tri komponente, konfiguracionih servera, delova (skupova replika) i rutera upita za sve klijente.

Za analizu i poboljšanje performansi baze podataka koriste se različite metode kao što su profilisanje, uvođenje indeksa, korišćenje projekcije, ograničavanje broja vraćenih zapisa, kao i poboljšanja na hardverskom nivou.

MongoDB baza podataka ima ugrađene mehanizme za prevazilaženje i smanjenje zastoja do koga dolazi usled migracije instance baze podataka sa jednog servera na drugi.

Na kraju, nadgledanje je jedan od kritičnih zadataka administracije baze podataka. MongoDB ima mnogo alata za praćenje, profilisanje i dijagnostifikovanje baze podataka.

10. Literatura

- [1] MongoDB High Availability, Afshin Mehrabani, ISBN: 9781783986736, <https://www.scribd.com/book/272074313/MongoDB-High-Availability>, datum poslednjeg pristupa 17.09.2021.
- [2] Highly Available Data: Why High Availability Is Not Just for Apps, <https://www.precisely.com/blog/data-availability/high-availability-not-just-apps>, datum poslednjeg pristupa 16.09.2021.
- [3] high availability (HA), <https://searchdatacenter.techtarget.com/definition/high-availability>, datum poslednjeg pristupa 16.09.2021.
- [4] Skalabilni klaster algoritmi, Seminarski rad iz Istraživanja podataka, Maljković Mirjana, Smer Informatika, master studije, Matematički fakultet, Beograd, <http://alas.matf.bg.ac.rs/~mi05006/ip/seminarskiSkalabilniKlasterAlgoritmi.pdf>, datum poslednjeg pristupa 16.09.2021.
- [5] Replication, <https://docs.mongodb.com/manual/replication/>, datum poslednjeg pristupa 16.09.2021.
- [6] Replica Set Oplog, <https://docs.mongodb.com/manual/core/replica-set-oplog/>, datum poslednjeg pristupa 16.09.2021.
- [7] Profiling (computer programming), [https://en.wikipedia.org/wiki/Profiling_\(computer_programming\)](https://en.wikipedia.org/wiki/Profiling_(computer_programming)), datum poslednjeg pristupa 17.09.2021.