



UNIVERZITET U NIŠU  
ELEKTRONSKI FAKULTET  
Katedra za računarstvo



## **APACHE CASSANDRA – CLUSTER REŠENJA**

Predmet: Sistemi za upravljanje bazama podataka  
-Seminarski rad-

Student:

Marijana Cvetković, br. ind. 1431

Mentor:

Doc. dr Aleksandar Stanimirović

Niš, jun 2022. godina

# Sadržaj

1. Uvod .....	3
2. Klaster baze podataka .....	4
2.1. Funkcionisanje arhitekture klastera.....	5
3. Klaster rešenja kod Apache Cassandra baze podataka.....	6
3.1. Veličina Cassandra klastera .....	8
3.2. Inicijalizacija klastera.....	8
3.2.1. Inicijalizacija klastera sa više čvorova (jedan centar podataka).....	8
3.2.2. Inicijalizacija klastera sa više čvorova (više centara podataka).....	11
3.3. Dodavanje čvora u postojeći klaster .....	14
3.4. Zamena čvora.....	15
3.4.1. Zamena aktivnog čvora .....	15
3.4.2. Zamena mrtvog čvora ili mrtvog semenskog čvora.....	16
3.5. Uklanjanje čvora .....	17
3.6. Komunikacija čvorova u klasteru – Gossip protokol.....	18
3.6.1. Otkrivanje kvarova i oporavak.....	20
3.7. Provera statusa klastera Cassandre .....	20
4. Praktična primena .....	22
5. Zaključak .....	26
6. Literatura.....	27

# 1. Uvod

Grupisanje baza podataka se odnosi na sposobnost nekoliko servera ili instanci da se povežu na jednu bazu podataka. Instanca je kolekcija memorije i procesa koji stupaju u interakciju sa bazom podataka, što je skup fizičkih datoteka koje zapravo čuvaju podatke. Da bi se obezbedila topologija visoke dostupnosti, mogu se kreirati klasteri baze podataka na različitim računarima. Grupisanjem baza podataka kombinuje se računarska snaga uključenih servera da bi se obezbedila veća sklabilnost, više kombinovane računarske snage ili ugrađenu redundantnost koja obezbeđuje veću dostupnost informacija.

Glavni razlozi upotrebe klastera baze podataka su prednosti koje ona nudi i to su redundantnost podataka, balansiranje opterećenja, visoka dostupnost, nadzor i automatizacija. Grupisanje je definitivno korisno sposobnošću balansiranja opterećenja i visoke dostupnosti. Ako se jedan čvor sruši, zahtevom obrađuje drugi čvor. Shodno tome, postoji malo ili nimalo mogućnosti apsolutnih kvarova sistema.

U ovom radu je u drugom poglavlju opisan pojam klasterovanja i koje su njegove prednosti, dati su neki tipovi klastera i u kojim slučajevima se oni koriste.

Treće poglavlje sadrži detaljan opis upotrebe klaster rešenja kod Apache Cassandra baze podataka. Detaljno je objašnjen postupak inicijalizacije klastera, sa jednim ili više centara, date operacije koje se koriste u radu sa čvorovima u klasteru, kao što su dodavanje novog čvora, zamena čvora i brisanje čvora. U posebnom poglavlju opisan je Gossip protokol koji se koristi za komunikaciju među čvorovima klastera. Navedena je upotreba ovog protokola za oporavak i otkrivanje kvarova unutar nekog klastera.

U četvrtom poglavlju nalazi se deo koji se tiče praktične primene klastera. Klaster je pokrenut na dva čvora (računar i laptop) i prikazan je ceo postupak inicijalizacije, konfiguracije i uspešnog pokretanja. U bazu podataka su uneti podaci, gde se vidi da oba čvora mogu pristupiti podacima.

Na kraju, u poslednjem poglavlju se nalazi zaključak izveden na osnovu obrađenog teorijskog dela i praktične primene ove teme.

## 2. Klaster baze podataka

Klaster baze podataka je kolekcija baza podataka kojima upravlja jedna instanca pokrenutog servera baze podataka. Nakon inicijalizacije, klaster baze podataka će sadržati bazu podataka pod nazivom postgres, koja je zamišljena kao podrazumevana baza podataka koji koriste uslužni programi, korisnici i aplikacije trećih strana.

Grupisanje baza podataka je proces kombinovanja više od jednog servera ili instanci koje povezuju jednu bazu podataka. Ponekad jedan server možda nije adekvatan za upravljanje količinom podataka ili brojem zahteva, tada je potreban klaster podataka. Grupisanje baze podataka, klasterisanje SQL servera i SQL klasterisanje su usko povezani sa SQL jezikom koji se koristi za upravljanje informacijama baze podataka.

Glavni razlozi za klasterisanje baze podataka su prednosti koje server dobija, redudantnost podataka, balansiranje opterećenja, visoka dostupnost, i na kraju, nadzor i automatizacija:

- **Redudansa podataka**

Više računara radi zajedno da skladišti podatke jedni među drugima uz grupisanje baze podataka. Ovo daje prednost redudantnosti podataka. Svi računari su sinhronizovani što znači da će svaki čvor imati potpuno iste podatke kao i ostali čvorovi. U bazi podataka treba izbegavati vrste ponavljanja (suvišnosti) koje dovode do dvosmislenosti podataka. Vrsta redudancije koju nudi klasterisanje je izvesna zbog sinhronizacije. U slučaju kvara na računaru, svi podaci će biti dostupni drugima.

- **Balansiranje opterećenja**

Balansiranje opterećenja ili skalabilnost ne dolazi po podrazumevanoj vrednosti sa bazom podataka. Mora se redovno donositi grupisanjem. Takođe zavisi od podešavanja. U osnovi, ono što radi balansiranje opterećenja je dodeljivanje radnog opterećenja između različitih računara koji su deo klastera. Ovo ukazuje da se može podržati više korisnika i ako se iz nekih razloga pojavi veliki skok u saobraćaju, postoji veća sigurnost da će moći da podrži novi saobraćaj. Jedna mašina neće postići sve pogotke. Ovo može da obezbedi neprimetno skaliranje po potrebi. Ovo se direktno povezuje sa visokom dostupnošću. Bez balansiranja opterećenja, određena mašina bi mogla biti preopterećena i saobraćaj bi se usporio, što bi dovelo do smanjenja saobraćaja na nulu.

- **Visoka dostupnost**

Kada se može pristupiti bazi podataka, to znači da je dostupna. Visoka dostupnost se odnosi na količinu vremena u kojoj se baza podataka smatra dostupnom. Količina dostupnosti koja je potrebna u velikoj meri zavisi od broja transakcija koje se pokreću u bazi podataka i koliko često se pokreće bilo koja vrsta analitike za podatke. Uz klasterisanje baze podataka, može se dostići izuzetno visok nivo dostupnosti zbog balansiranja opterećenja i imati dodatne mašine. Međutim, u slučaju da se server isključi, baza podataka biće dostupna.

- **Monitoring i automatizacija**

Za ovaj zadatak se može koristiti normalna baza podataka jer se praćenje i automatizacija mogu lako obaviti pomoću softvera. Prednost postaje očiglednija kada je prisutan klaster. Tipično, prednost je u tome što grupisanje omogućava automatizaciju mnogih procesa baze podataka u isto vreme što dozvoljava postavljanje pravila koja upozoravaju na potencijalne probleme. Ovo sprečava povratak da bi se sve proverilo ručno. Sa klasterizovanom bazom podataka, automatizacija je od pomoći jer će omogućiti dobijanje obaveštenja ako se sistem previše zahteva. Međutim, klaster će imati određenu mašinu koja će se koristiti kao sistem za upravljanje bazom

podataka/kontrolna tabla za ceo klaster. Ova izabrana mašina može imati skripte koje se pokreću automatski za ceo klaster baze podataka i rade sa svim čvorovima baze podataka. [1]

## 2.1. Funkcionisanje arhitekture klastera

U arhitekturi klastera, svi zahtevi su podeljeni sa mnogo računara tako da se pojedninačni zahtev korisnika izvršava i proizvodi od strane više računarskih sistema. Grupisanje je definitivno korisno sposobnošću balansiranja opterećenja i visoke dostupnosti. Ako se jedan čvor sruši, zahtevom obrađuje drugi čvor. Shodno tome, postoji malo ili nimalo mogućnosti apsolutnih kvarova sistema.

### Tipovi klastera baza podataka

Klaster baze podataka je veoma sveobuhvatan. Pokriva više nivoa i aranžmana u zavisnosti od zahteva sistema. Biće predstavljena tri tipa arhitektura klaster računara. Klasteri za napuštanje greške, klasteri visokih performansi i klasteri za balansiranje opterećenja.

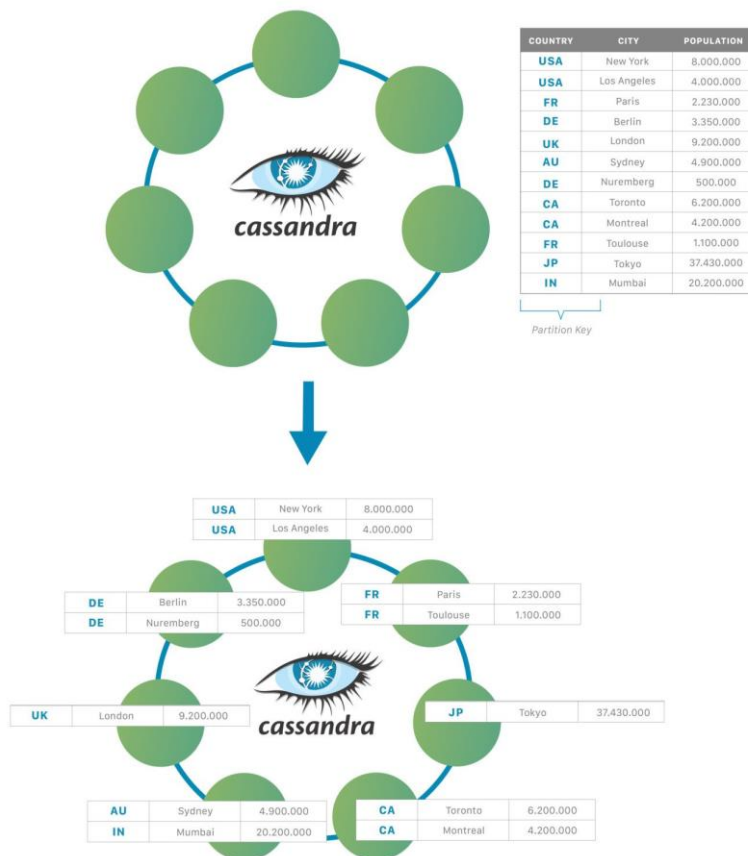
- **Klasteri za grešku/visokoku dostupnost:** Mašina može da pođe po zlu ili prestane da radi u bilo kom trenutku. Administratori sistema upravljaju takvim prelaskom na greške i efikasno rešavaju probleme. Ovde klasteri priskaču u pomoć. Klaster priprema dostupnost usluge repliciranjem servera i redudansom rekonfiguracijom softvera i hardvera. Dakle, svaki sistem kontroliše drugi i radi na zahtevima ako bilo koji čvor otkaže. Ove vrste klastera su profitabilne za one korisnike koji u potpunosti zavise od svojih računarskih sistema. Na primer, e-trgovina, veb stranice itd.  
Sistem treba da bude dovoljno sposoban da zna koji sve sistemi rade, sa koje IP adrese, koji zahtev i kakav bi bio napredak akcije u slučaju pada. Važno je da serveri ionako ne prestanu da rade.
- **Klasteri visokih performansi:** Svrha razvoja klastera baza podataka visokih performansi je proizvodnja računarskih sistema visokih performansi. Oni upravljaju ko-proširujućim programima koji su potrebni za dugotrajne proračune. Ovakvu raznolikost klastera obično preferiraju naučne industrije. Osnovni cilj je inteligentno podeliti opterećenje.
- **Klasteri za balansiranje opterećenja:** Ovi klasteri baze podataka služe za distribuciju opterećenja između različitih servera. Oni nastoje da obezbede povećan kapacitet mreže, konačno povećavajući performanse. Sistemi u ovoj mreži integrišu svoje čvorove, uz pomoć kojih se zahtevi korisnika podjednako dele na čvorove koji učestvuju. Sistem ne radi zajedno, već preusmerava zahteve podjednako kako se pojave.

Uprkos svim ovim distribucijama tehnologije u pozadini, čini se da je to jedinstven sistem za korisnika. Upotreba klastera varira od preduzeća, u zavisnosti od vrste procesa i zahtevanog nivoa performansi. [1]

### 3. Klaster rešenja kod Apache Cassandra baze podataka

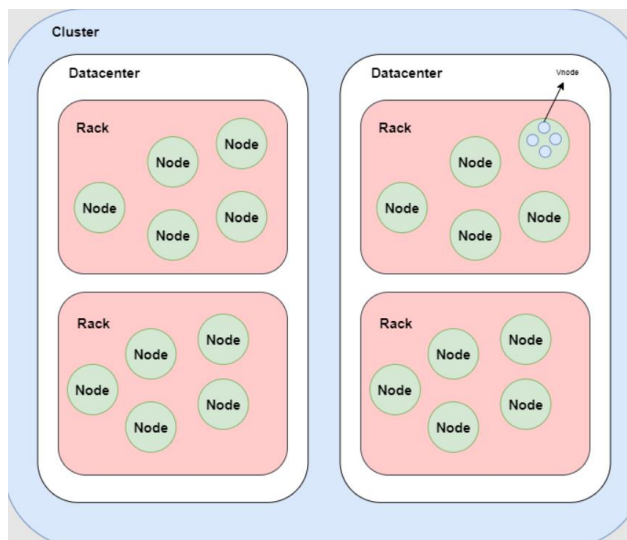
Apache Cassandra je NoSQL, distribuirani sistem za upravljanje bazom podataka. Glavna prednost Cassandre je u tome što može da rukuje velikom količinom struktuiranih podataka preko robnih servera. Cassandra obezbeđuje visoku dostupnost i ne obezbeđuje jednu tačku kvara. Cassandra to postže korišćenjem arhitekture prstenastog tipa, gde je najamnja logička jedinica čvor. Koristeći particionisanje podataka za optimizaciju upita.

Svaki deo podataka ima ključ za particiju. Ključ particije za svaki red je heširan. Kao rezultat, dobiće se jedinstveni token za svaki podatak. Za svaki čvor postoji dodeljeni opseg tokena. Shodno tome, podaci sa istim tokenom se čuvaju na istom čvoru. Arhitektura prstena je prikana na slici:



Slika: Arhitektura prstena kod Cassandre

Klaster je komponenta koja sadrži jedan ili više centara podataka. To je najspoljnji kontejner za skladištenje u bazi podataka. Jedna baza podataka sadrži jedan ili više klastera. Hijerarhija elemenata u klasteru Cassandra je prikazana na slici:

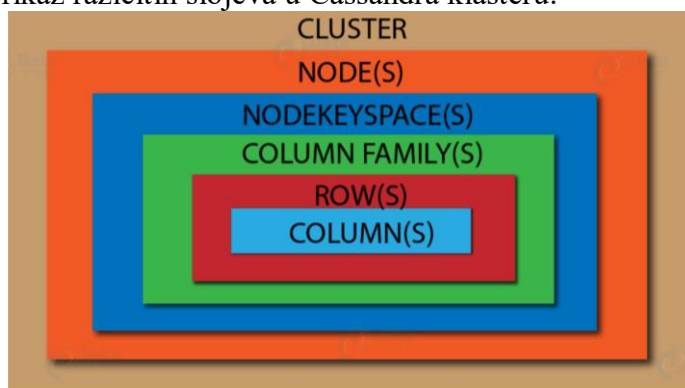


Slika: Klaster kod Cassandra-e

Prvo, imamo klaster koji se sastoji od centara podataka. Unutar centara podataka imamo čvorove koji podrazumevano sadrže 256 virtuelnih čvorova.

Čvor je osnovna infrastrukturna komponenta Cassandre. To je potpuno funkcionalna mašina koja se povezuje sa drugim čvorovima u klasteru preko visoke interne mreže. Svaki klaster može da sadrži mnogo čvorova ili sistema. Svaki čvor je nezavisan i ima istu ulogu u prstenu i svaki čvor će imati potrebnu procesorsku snagu (CPU), memoriju (RAM) i skladište (obično na trenutnim serverima, u obliku SSD uređaja, poznatih kao SSD). Cassandra raspoređuje čvorove u peer-to-peer strukturu. Čvor sadrži stvarne podatke. Svi čvorovi u klasteru mogu prihvatati zahteve za čitanje i pisanje. Stoga, nije važno gde se podaci zapravo nalaze u klasteru. Uvek će se dobiti najnovija verzija podataka. [2]

Novije verzije Cassandre koriste virtualne čvorove ili skraćeno vnode. Virtualni čvor je sloj za skladištenje podataka unutar servera. Podrazumevano postoji 256 virtualnih čvorova po serveru. Virtualni čvorovi pružaju veću fleksibilnost u sistemu. Cassandri je lakše da dodaje nove čvorove u klaster kada zatreba. Kada podaci imaju nejednako raspoređene tokene među čvorovima, lako može da se proširi kapacitet skladištenja tako što će se proširiti virtualni čvorovi na više opterećeni čvor. Na slici je dat prikaz različitih slojeva u Cassandra klasteru:



Slika: Slojevi u Cassandra klasteru [4]

Keyspace je sledeći sloj skladišta. U čvoru postoji mnogo ključeva. Ovi ključevi su u osnovi najudaljenija jedinica za skladištenje u sistemu. Oni sadrže glavne podatke. Podaci su raspoređeni prema njihovim svojstvima ili oblastima. Sloj ispod su porodice kolona (column familys). Prostor ključeva je dalje podeljen na porodice kolona. Ove porodice kolona imaju različite oblasti ili naslove pod kojima se distribuiraju podaci. U ključnom prostoru, ove porodice kolona su kategorisane u različite naslove. Ovi naslovi dalje sadrže različite slojeve skladištenih jedinica.

Sledeći sloj u bazi podataka su redovi prema porodicama kolona. Redovi su u osnovi klasifikacija prema kojoj je porodica kolona podeljana. Ove klasifikacije, zauzvrat, stvaraju specifične kriterijume distribucije za unose. Najdublji sloj u bazi podataka je kolona. Kolona je u osnovi podeljena na različite naslove. Ovi naslovi sadrže glavne podatke u vezi sa određenim unosom. [4]

Kada se koristi termin server, misli se na mašinu sa instaliranim Cassandra softverom. Svaki čvor ima jednu instancu Cassandre, koja je tehnički server. Cassandra server pokreće osnovne procese. Na primer, procesi poput širenja replika oko čvorova ili zahteva za rutiranje.

Cassandra koristi mehanizam koji se zove replikacija sa više centara podataka kako bi osigurala da se podaci prenose i na kraju sinhronizuju između svojih klastera.

### **3.1. Veličina Cassandra klastera**

Širenje čvorova Cassandra klastera predstavlja problem u današnjim centrima podataka. Takođe je poznato kao 'rasipano prekomereno obezbeđivanje', širenje Cassandra čvorova često odražava napor da se postigne niska latencija i visoka dostupnost. Često, sva prekomerna potrošnja na prekomerno obezbeđivanje ne rezulta ispunjenjem zahteva za performansama. Prekomerno obezbeđivanje može da obezbedi izvestan nivo jastuka (iako skup) protiv naglih skokova u saobraćaju, prekidu rada i drugih problema. A i ovo dolazi po cenu više grešaka na serveru i većih administrativnih troškova. [5]

### **3.2. Inicijalizacija klastera**

Postoje dva scenarija, Apache Cassandra klaster sa jednim centrom podataka i Cassandra klaster sa više centara podataka, u poglavljima 3.1.1. i 3.1.2. biće detaljno opisana oba.

#### **3.2.1. Inicijalizacija klastera sa više čvorova (jedan centar podataka)**

U ovom poglavlju biće reči o primeni klastera Apache Cassandre sa jednim centrom podataka. Postoje preduslovi koji treba da budu ispunjeni kako bi moglo da se koristi. Svaki čvor mora biti ispravno konfigurisan pre pokretanja klastera. Koraci koje je potrebno odrediti ili izvršiti pre pokretanja klastera su:

- Dobro razumevanje kako Cassandra radi.
- Instaliranje Cassandre na svakom čvoru.
- Odabir imena za klaster.
- Dobijanje IP adrese svakog čvora.
- Određivanje koji čvorovi će biti semenski čvorovi. Ne prave se svi čvorovi semenskim čvorovima.



- Određivanje strategije dojavljivanja i replikacije. Preporučeni su `GossipingPropertyFileSnitch` i `NetworkTopologyStrategy`.
- Određivanje konvencije imenovanja za svaki stalak. Na primer, dobra imena su `RAC1`, `RAC2` ili `R101`, `R102`.
- Konfiguraciona datoteka `cassandra.yaml` i datoteke svojstava kao što je `cassandra-rackdc.properties` daju više opcija za konfiguraciju. [3]

Ovaj primer opisuje instaliranje klastera od 6 čvorova koji obuhvata 2 stalka u jednom centru podataka. Svaki čvor je već konfigurisan da koristi `GossipingPropertyFileSnitch` i 256 virtualnih čvorova (`vnodes`).

U Cassandri 'centar podataka' je sinonim za 'grupnu replikaciju'. Oba termina se odnose na skup čvorova konfigurisanih kao grupa za potrebe replikacije.

Procedura je sledeća:

(1) Pretpostavlja se da je Cassandra instalirana na sledećim čvorovima:

```
node0 110.82.155.0 (seed1)
node1 110.82.155.1
node2 110.82.155.2
node3 110.82.156.3 (seed2)
node4 110.82.156.4
node5 110.82.156.5
```

Najbolja praksa je da postoji više od jednog početnog čvora po centru podataka.

- (2) Ako postoji zaštitni zid koji radi u klasteru, potrebno je otvoriti određene portove za komunikaciju između čvorova.
- (3) Ako Cassandra radi, mora se zaustaviti server i izbrisati podaci. Osim uklanjanja podrazumevanog `cluster_name` (Test Cluster) iz sistemske tabele. Svi čvorovi mora da koriste isto ime klastera.

Instalacija paketa:

- Zaustavljanje Cassandre:

```
sudo service cassandra stop #Stops Cassandra
```

- Brisanje podataka:

```
sudo rm -rf /var/lib/cassandra/*
```

Instalacije tarball:

- Stopiranje Cassandre:
 

```
ps auxx | grep Cassandra
```

```
sudo kill pid
```
- Brisanje podataka:

```
sudo rm -rf install_location/data/*
```

(4) Podešavanje svojstva u datoteci cassandra.yaml za svaki čvor:

Nakon što se bilo koja izmena izvrši u datoteci cassandra.yaml, mora se ponovo pokrenuti čvor da bi promene stupile na snagu.

Svojstva za podešavanje su:

- cluster\_name:
- num\_tokens: recommended value: 256
- -seeds: interna IP adresa svakog seed čvora  
U novim klasterima. Početni čvorovi ne vrše pokretanje (proces pridruživanja novog čvora postojećem klasteru).
- listen\_address:  
Ako je čvor početni čvor, ova adresa mora da se poklapa sa IP adresom na listi za početak. U suprotnom, komunikacija tračeva propada jer se ne zna seme.  
Ako nije podešeno, Cassandra traži od sistema lokalnu adresu, onu koja je povezana sa njegovim imenom hosta. U nekim slučajevima Cassandra ne daje tačnu adresu i mora se navesti adresa saslušanja.
- rpc\_address: slušajte adresu za klijentske veze
- endpoint\_snitch: ime doušnika

Ako su čvorovi u klasteru identični u pogledu izgleda diska, deljenih biblioteka i tako dalje, može se koristiti ista datoteka cassandra.yaml za sve njih.

Primer:

```
cluster_name: 'MyCassandraCluster'
num_tokens: 256
seed_provider:
  - class_name: org.apache.cassandra.locator.SimpleSeedProvider
    parameters:
      - seeds: "110.82.155.0,110.82.155.3"
listen_address:
rpc_address: 0.0.0.0
endpoint_snitch: GossipingPropertyFileSnitch
```

Ako je rpc\_address podešena na džoker adresu (0.0.0.0), onda broadcast\_rpc\_address mora biti podešena ili usluga neće ni da se pokrene.

(5) U datoteci cassandra-rackdc.properties dodeljuju se nazivi centara podataka i rack-a koje su određeni u preduslovima. Na primer:

```
# indicate the rack and dc for this node
dc=DC1
rack=RAC1
```

(6) GossipingPropertyFileSnitch uvek učitava cassandra-topology.properties kada je ta datoteka prisutna. Uklanja se datoteka sa svakog čvora na bilo kom novom klasteru ili bilo kom klasteru koji je migriran iz PropertyFileSnitch.

- (7) Nakon što je instalirana i konfigurisana Cassandra na svim čvorovima, DataStak preporučuje pokretanje početnih čvorova jedan po jedan, a zatim pokretanje ostalih čvorova.

Ako se čvor ponovo pokrenuo zbog automatskog ponovnog pokretanja, prvo se mora zaustaviti čvor i obrisati direktorijumu, kao što je već opisano.

Instaliranje paketa:

```
sudo service cassandra start #Starts Cassandra
```

Tarball instalacije:

```
cd install_location
```

```
bin/cassandra
```

- (8) Da bi se proverilo da li je prsten pokrenut i radi, pokreće se:

Instalacija paketa:

```
nodetool status
```

Tarball instalacija:

```
cd install_location
```

```
bin/nodetool status
```

Izlaz treba da navede svaki čvor i da prikaže njegov status kao UN (Up Normal), kao na slici:

```
paul@ubuntu:~/cassandra-2.1.0$ bin/nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
// State=Normal/Leaving/Joining/Moving
-- Address          Load          Tokens      Owns    Host ID                               Rack
UN  10.194.171.160    53.98 KB      256        0.8%    a9fa31c7-f3c0-44d1-b8e7-a2628867840c rack1
UN  10.196.14.48      93.62 KB      256        9.9%    f5bb146c-db51-475c-a44f-9facf2f1ad6e rack1
UN  10.196.14.239     83.98 KB      256        8.2%    b8e6748f-ec11-410d-c94f-b8e7d88a28e7 rack1
...
```

Slika: Izlaz

### 3.2.2. Inizijalizacija klastera sa više čvorova (više centara podataka)

U ovom poglavlju biće opisana primena klastera Apache Cassandra sa više centara podataka. Ovaj primer opisuje instaliranje klastera sa šest čvorova koji obuhvata dva centra podataka. Svaki čvor je konfigurisan da koristi GossipingPropertyFileSnitch i 256 virtualnih čvorova (vnodes). U Cassandri 'centar podataka' je sinonim za 'grupu za replikaciju'. Oba termina se odnose na skup čvorova konfigurisanih kao grupa za potrebe replikacije.

Svaki čvor mora biti ispravno konfigurisan pre pokretanja klastera. Moraju se odrediti ili izvršiti sledeće stavke pre pokretanja klastera:

- Dobro razumevanje kako Cassandra radi.
- Instaliranje Cassandre na svakom čvoru.
- Odabir imena za klaster.
- Dobijanje IP adrese svakog čvora.
- Određivanje koji čvorovi će biti semenski čvorovi. Ne prave se svi čvorovi semenskim čvorovima.
- Određivanje strategije dojavljivanja i replikacije. Preporučeni su GossipingPropertyFileSnitch i NetworkTopologyStrategy.
- Određivanje konvencije imenovanja za svaki stalak. Na primer, dobra imena su RAC1, RAC2 ili R101, R102.
- Konfiguraciona datoteka *cassandra.yaml* i datoteke svojstava kao što je *cassandra-rackdc.properties* daju više opcija za konfiguraciju.

Nakon ispunjenja prethodnih preduslova, procedura je sledeća:

- (1) Pretpostavka da je Cassandra instalirana na sledećim čvorovima:

```
node0 10.168.66.41 (seed1)
node1 10.176.43.66
node2 10.168.247.41
node3 10.176.170.59 (seed2)
node4 10.169.61.170
node5 10.169.30.138
```

Najbolja praksa je imati više od jednog početnog čvora po centru podataka.

- (2) Ako postoji zaštitni zid koji radi u klasteru, potrebno je otvoriti određene portove za komunikaciju između čvorova.
- (3) Ako Cassandra radi, mora se zaustaviti server i izbrisati podaci. Osim uklanjanja podrazumevanog cluster\_name (Test Cluster) iz sistemske tabele. Svi čvorovi mora da koriste isto ime klastera.

Instalacija paketa:

- Zaustavljanje Cassandre:

```
sudo service cassandra stop #Stops Cassandra
```

- Brisanje podataka:

```
sudo rm -rf /var/lib/cassandra/*
```

Instalacije tarball:

- Stopiranje Cassandre:

```
ps aux | grep Cassandra
```

```
sudo kill pid
```

- Brisanje podataka:

```
sudo rm -rf install_location/data/*
```

- (4) Podešavanje svojstva u datoteci *cassandra.yaml* za svaki čvor:

Nakon što se bilo koja izmena izvrši u datoteci *cassandra.yaml*, mora se ponovo pokrenuti čvor da bi promene stupile na snagu.

Svojstva za podešavanje su:

- `cluster_name`:
- `num_tokens`: recommended value: 256
- `-seeds`: interna IP adresa svakod seed čvora  
U novim klasterima. Početni čvorovi ne vrše pokretanje (proces pridruživanja novog čvora postojećem klasteru).
- `listen_address`:  
Ako je čvor početni čvor, ova adresa mora da se poklapa sa IP adresom na listi za početak. U suprotnom, komunikacija tračeva propada jer se ne zna seme.  
Ako nije podešeno, Cassandra traži od sistema lokalnu adresu, onu koja je povezana sa njegovim imenom hosta. U nekim slučajevima Cassandra ne daje tačnu adresu i mora se navesti adresa saslušanja.
- `rpc_address`: slušajte adresu za klijentske veze
- `endpoint_snitch`: ime doušnika

Ako su čvorovi u klasteru identični u pogledu izgleda diska, deljenih biblioteka i tako dalje, može se koristiti ista datoteka `cassandra.yaml` za sve njih.

Primer:

```
cluster_name: 'MyCassandraCluster'
num_tokens: 256
seed_provider:
  - class_name: org.apache.cassandra.locator.SimpleSeedProvider
    parameters:
      - seeds: "10.168.66.41,10.176.170.59"
listen_address:
endpoint_snitch: GossipingPropertyFileSnitch
```

- (5) U datoteci `cassandra-rackdc.properties` dodeljuju se nazivi centara podataka i rack-a koje su određeni u preduslovima. Na primer:

Čvorovi 0 do 2:

```
## Indicate the rack and dc for this node
dc=DC1
rack=RAC1
```

Čvorovi 3 do 5:

```
## Indicate the rack and dc for this node
dc=DC2
rack=RAC1
```

(6) GossipingPropertyFileSnitch uvek učitava cassandra-topology.properties kada je ta datoteka prisutna. Uklanja se datoteka sa svakog čvora na bilo kom novom klasteru ili bilo kom klasteru koji je migriran iz PropertyFileSnitch.

(7) Nakon što je instalirana i konfigurisana Cassandra na svim čvorovima, DataStak preporučuje pokretanje početnih čvorova jedan po jedan, a zatim pokretanje ostalih čvorova.

Ako se čvor ponovo pokrenuo zbog automatskog ponovnog pokretanja, prvo se mora zaustaviti čvor i obrisati direktorijumu, kao što je već opisano.

Instaliranje paketa:

```
sudo service cassandra start #Starts Cassandra
```

Tarball instalacije:

```
cd install_location
```

```
bin/cassandra
```

(8) Da bi se proverilo da li je prsten pokrenut i radi, pokreće se:

Instalacija paketa:

```
nodetool status
```

Tarball instalacija:

```
cd install_location
```

```
bin/nodetool status
```

Izlaz treba da navede svaki čvor i da prikaže njegov status kao UN (Up Normal), kao na slici:

```
paul@ubuntu:~/cassandra-2.1.0$ bin/nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
// State=Normal/Leaving/Joining/Moving
--  Address                Load       Tokens     Owns    Host ID                               Rack
UN  10.194.171.160          53.98 KB   256        0.8%    a9fa31c7-f3c0-44d1-b8e7-a2628867840c rack1
UN  10.196.14.48             93.62 KB   256        9.9%    f5bb146c-db51-475c-a44f-9facf2f1ad6e rack1
UN  10.196.14.239           83.98 KB   256        8.2%    b8e6748f-ec11-410d-c94f-b8e7d88a28e7 rack1
...
```

Slika: Izlaz

### 3.3. Dodavanje čvora u postojeći klaster

Virtuelni čvorovi (vnodes) u velikoj meri pojednostavljaju dodavanje čvorova u postojeći klaster:

- Više nije potrebno izračunavanje tokena i njihovo dodeljivanje svakom čvoru.
- Ponovno balansiranje klastera više nije neophodno jer čvor koji se pridružuje klasteru preuzima odgovornost za paran deo podataka.

Neophodno je koristiti istu verziju Cassandre na svim čvorovima u klasteru.

Procedura za dodavanje je sledeća:

- (1) Instaliranje Cassandra na novim čvorovima, ali ne treba je pokretati. Ako se instalacija Cassandra na Debianu pokrene automatski, mora se zaustaviti čvr i izbrisati podaci.
- (2) U zavisnosti od dodeljivanja koji se koristi u klasteru, podešavaju se svojstva u datoteci `cassandra-topology.properties` ili `cassandra-rackdc.properties`:
  - `PropertyFileSnitch` koristi datoteku `cassandra-topology.properties`.
  - `GossipingPropertyFileSnitch`, `Ec2Snitch`, `Ec2MultiRegionSnitch` i `GoogleCloudSnitch` koriste datoteku `cassandra-rackdc.properties`.
- (3) Potrebno je podesiti sledeća svojstva u datoteci `cassandra.yaml`:
  - `auto_bootstrap` – Ako je ova opcija postavljena na `false`, mora se postaviti na `true`. Ova opcija nije navedena u podrazumevanoj konfiguracionoj datoteci `cassandra.yaml` i podrazumevano je postavljena na `true`.
  - `cluster_name` – Ime klastera kojem se pridružuje novi čvor.
  - `listen_address/broadcast_address` – Obično se može ostaviti prazno. U suprotnom, koristi se IP adresa ili ime hosta koje drugi Cassandra čvorovi koriste za povezivanje sa novim čvorom.
  - `endpoint_snitch` – Doušnik koji Cassandra koristi za lociranje čvorova i zahteve za rutiranje.
  - `num_tokens` – Broj vnodova koje treba dodeliti čvoru. Koristi se isti broj tokena koji je postavljen i na drugim čvorovima u centru podataka. Opsezi tokena su proporcionalno raspoređeni, ako se hardverske mogućnosti razlikuju, dodeljuju se više opsega tokena sistemima sa većim kapacitetom i boljim performansama.
  - `allocate_tokens_for_local_replication_factor` – Navodi se faktor replikacije prostora ključeva u centru podataka.
  - `seed_provider` – Lista -seeds određuje koje čvorove novi čvor trebda da kontaktira da bi saznao više o klasteru i uspostavio proces ogovaranja.
- (4) Pokretanje bootstrap čvora.
- (5) Uz pomoć status `nodetool`-a vrši se provera da li je čvor potpuno pokrenut i da su svi ostali čvorovi podignuti (UN) i da nisu u bilo kom drugom stanju.
- (6) Nakon što su pokrenuti svi čvorovi, pokreće se `nodetool cleanup` na svakom od prethodno postojećih čvorova da bi se uklonili ključevi koji više ne pripadaju tim čvorovima. Potrebno je sačekati da se čišćenje završi na jednom čvoru pre nego što se pokrene čišćenje alata `node` na sledećem čvoru. [7]

### **3.4. Zamena čvora**

Zamena čvora može biti neophodna iz više razloga, te je neophodno znati postupak. Zamena se razlikuje kod zamena aktivnog čvora i zamene mrtvog čvora, što će biti predstavljeno u poglavjima 3.4.1. i 3.4.2.

#### **3.4.1. Zamena aktivnog čvora**

Koraci za zamenu čvora novim čvorom, na primer prilikom ažuriranja na noviji hardver ili proaktivnog održavanja nalaze se u ovom poglavlju. Radni čvor se može zameniti na dva načina:

- Dodavanje čvora, a zatim povlačenje starog čvora.
- Korišćenje `nodetool`-a za zamenu pokrenutog čvora.

Da bi se promenila IP adresa čvora, jednostavno se menja IP adresa čvora, a zatim ponovo pokreće Cassandra. Ako se promeni IP adresa početnog čvora, mora se ažurirati parametar -seeds na listi seed\_provider u datoteci cassandra.yaml svakog čvora.

Kod dodavanja čvora, a onda povlačenja starog, mora se prvo pripremiti i pokrenuti zamenski čvor, integrirati ga u klaster, a zatim poništiti stari čvor. Procedura je sledeća:

- (1) Priprema i pokretanje zamenskog čvora.
- (2) Potvrda da je zamenski čvor živ (pokretanje nodetool ring ako se ne koristi vnodes ili pokretanje status nodetool ako se koristi vnodes) i status treba da pokaže nodetool ring:UP ili status nodetool:UN.
- (3) Treba znati ID domaćina originalnog čvora, za potrebe sledećeg koraka.
- (4) Korišćenjem ID hosta originalnog čvora, poništava se rad originalnog čvora iz klastera pomoću komande nodetool za dekomicioniranje.
- (5) Pokretanje nodetool čišćenje na svim ostalim čvorovima u istom centru podataka.

Kod korišćenja nodetool-a za zamenu pokrenutog čvora, omogućeno je da se zameni pokrenuti čvor dok se izbegava dva puta strimovanje podataka ili pokretanje čišćenja. Treba obratiti pažnju da ako se koristi nivo doslednosti ONE, rizikuje se da se izgube podaci jer čvor može da sadrži jedinu kopiju zapisa. Potputno treba biti siguran da nijedna aplikacija ne koristi nivo doslednosti ONE.

Procedura ovog pristupa je sledeća:

- (1) Zaustavljanje Cassandre a čvoru koji se treba zameniti.
- (2) Praćenje uputstva za zamenu mrtvog čvora koristeći IP adresu starog čvora za – Dcassandra.replace\_address.
- (3) Uveriti se da se nivo doslednosti ONE ne koristi na ovom čvoru.
- (4) Uklanjanje čvora. [9]

### 3.4.2. Zamena mrtvog čvora ili mrtvog semenskog čvora

U ovom poglavlju se nalaze koraci za zamenu čvora koji je umro iz nekog razloga, kao što je recimo kvar hardvera. Procedura za zamenu mrtvog čvora je ista za vnode i čvorove sa jednim tokenom. Potrebni su dodatni koraci za zamenu mrtvih semenskih čvorova.

Procedura:

- (1) Pokretanje status nodetool da bi se potvrdilo da je čvor mrtav (DN), kao što se vidi na slici:

```
paul@ubuntu:~/cassandra-2.1.0$ bin/nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
// State=Normal/Leaving/Joining/Moving
-- Address                      Load          Tokens      Owns    Host ID                               Rack
UN  10.194.171.160                53.98 KB      256         0.8%    a9fa31c7-f3c0-44d1-b8e7-a2628867840c rack1
UN  10.196.14.48                   93.62 KB      256         9.9%    f5bb146c-db51-475c-a44f-9facf2f1ad6e rack1
DN  10.196.14.239                  ?             256         8.2%    null
```

Slika: Status mrtvog čvora



- (2) Beleženje podešavanja centra podataka, adrese i stalka mrtvog čvora, korišćiće se kasnije.
- (3) Dodavanje zamenskog čvora u mrežu i snimanje njegove IP adrese.
- (4) Ako je mrtvi čvor bio početni čvor, menja se konfiguracija početnog čvora klastera na svakom čvoru:
  - U datoteci `cassandra.yaml` za svaki čvor, uklanja se IP adresa mrtvog čvora sa `-seeds` liste u svojstvu dobavljača semena.
  - Ako klasteru treba novi početni čvor da zameni mrtvi čvor, dodaje se IP adresa novog čvora na `-seeds` listu.
  - Ponovno pokretanje čvora.
- (5) Na postojećem čvoru, prikupe se informacije o podešavanjima za nov čvor iz datoteke `cassandra.yaml`:
  - `Cluster_name`
  - `Endpoint_snitch`
  - Ostala podešavanja koja nisu podrazumevana.
- (6) Prikupljanje informacija o racku i centru podataka.
- (7) Uveravanje da novi čvor ispunjava sve preduslove i zatim se instalira Cassandra na novi čvor, ali bez pokretanja.
- (8) Ako se Cassandra automatski pokreće na čvoru, zaustavlja se i brišu podaci koji su automatski dodati pri pokretanju.
- (9) Dodavanje vrednosti sledećim svojstvima u `cassandra.yaml` fajlu iz informacija koje su prikupljene ranije: `auto_bootstrap(true)`, `cluster_name` i spisak semena.
- (10) Dodavanje konfiguracije rack i data centra.
- (11) Pokretanje novog čvora sa opcijom `replace_address`, prenoseći IP adresu mrtvog čvora.
- (12) Pokretanje status `nodetool` da bi se proverilo da je novi čvor uspešno pokrenut. [9]

### 3.5. Uklanjanje čvora

Ukoliko je potrebno ukloniti neki od čvorova kako bi se smanjila veličina klastera, procedura je sledeća:

- (1) Provera da li je čvor UP ili DOWN koristeći `nodetool` status. Komanda `nodetool` prikazuje status čvora (slika ):

```
paul@ubuntu:~/cassandra-1.2.0$ bin/nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address          Load          Tokens   Owns    Host ID                               Rack
UN  10.194.171.160    53.98 KB      256      0.8%    a9fa31c7-f3c0-44d1-b8e7-a2628867840c rack1
UN  10.196.14.48      93.62 KB      256      9.9%    f5bb146c-db51-475c-a44f-9facf2f1ad6e rack1
DN  10.196.14.239     ?             256      8.2%    null
```

Slika: Status čvora

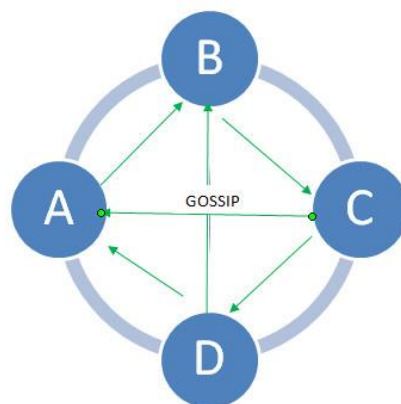
- (2) Ako je čvor podignut, pokreće se `nodetool decommission`. Ovo dodeljuje opsege za koje je čvor bio odgovoran drugim čvorovima i na odgovarajući način replicira podatke.
- (3) Ako čvor ne radi, bira se odgovarajuća opcija:
  - Ako klaster koristi `vnode`, uklanja se čvor pomoću naredbe `nodetool removemode`.

- Ako klaster ne koristi vnode, pre pokretanja komande *nodetool removenode*, prilagođavaju se tokeni da ravnomerno rasporede podatke na preostale čvorove kako bi se izbeglo stvaranje vruće tačke.
- (4) Ako *removenode* ne uspe, pokreće se *nodetool assassinate*. To je prisilno uklanjanje mrtvih čvorova bez ponovnog repliciranja podataka. To je poslednje sredstvo ako se ne može uspešno ukloniti *nodetool removenode*. [9]

### 3.6. Komunikacija čvorova u klasteru – Gossip protokol

Unutar Cassandra klastera ne postoji centralni primarni (ili glavni) čvor. Svi čvorovi u klasteru su ravnopravni. Postoje mehanizmi, kao što je Gossip protokol, koji određuje kada se klaster prvi put pokreće kako bi čvorovi otkrili jedni druge. Međutim, kada je topologija uspostavljena, ona nije statična. Ovaj isti mehanizam Gossip pomaže da se utvrdi kada se dodaju dodatni čvorovi u klaster, ili kada se čvorovi uklone iz klastera. [5]

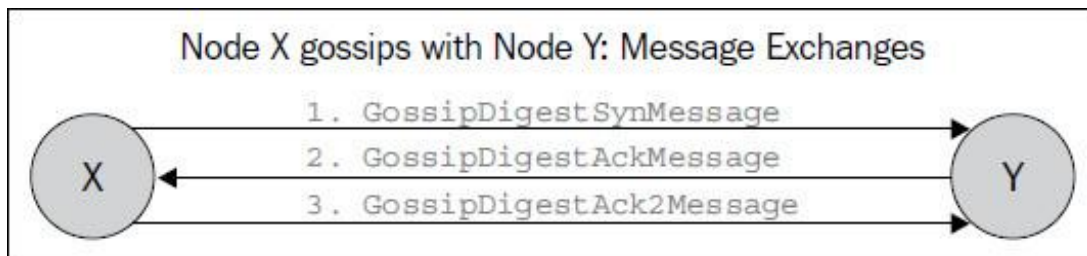
Gossip protokol je tehnika interne komunikacije za čvorove u klasteru da razgovaraju jedni sa drugima. Gossip je efikasan, lagan, pouzdan inter-nodalni protokol emitovanja za širenje podataka. Decentralizovan je 'epidemijski', tolerantan je na greške i protokol za međusobnu komunikaciju. Cassandra koristi ogovaranje za otkrivanje kolega i širenje metapodataka. Na slici je prikazan rad ovog protokola: [6]



PEER TO PEER DISTRIBUTION  
MODEL OF CASSANDRA

Slika: Gossip

Proces ogovaranja se pokreće svake sekunde za svaki čvor i razmenjuje poruke u stanju sa do tri druga čvora u klasteru. Pošto je ceo proces decentralizovan, ne postoji ništa ili niko ko koordinira svaki čvor za ogovaranje. Svaki čvor nezavisno će uvek izabrati jednog do tri vršnjaka sa kojima će ogovarati. Uvek će izabrati živog peer-a (ako ga ima) u klasteru, verovatno će izabrati početni čvor iz klastera ili će možda verovatno izabrati nedostupan čvor.



Slika:

Gossip protokol je veoma slična TCP trosmernom rukovanju. Sa redovnim protokolom emitovanja, mogla je biti samo jedna poruka po rundi, a podacima se može dozvoliti da se postepeno šire kroz klaster. Ali sa protokolom ogovaranja, imati tri poruke za svaki krug ogovaranja daje stepen anantropije. Ovaj proces omogućava mnogo brže 'konvergencije' podataka koji se dele između dva čvora u interakciji.

SYN: Čvor koji pokreće krug ogovaranja šalje SYN poruku koja sadrži kompendijum čvorova u klasteru. Sadrži nizove IP adrese čvora u klasteru, generaciju i berziju otkućaja čvora.

ACK: Peer nakon što primi SYN poruku upoređuje sopstvene informacije o metapodacima sa onima koje je poslao inicijator i proizvodi diff. ACK sadrži dve vrste podataka. Jedan deo se sastoji od ažuriranih informacija o metapodacima (AppStates) koje ravnopravni partner ima, ali inicijator nema, a drugi deo se sastoji od sažetka čvorova koje inicijator ima, a ne.

ACK2: Inicijator prima ACK od ravnopravnog partnera u ažurira njegove metapodatke od AppStates-a i šalje nazad ACK2 koji sadrži informacije o metapodacima koje je ravnopravni partner zatražio. Vršnjak prima ACK2, ažurira svoje metapodatke i krug tračeva se završava.

Ovde je važna napomena da će ovaj protokol za razmenu poruka izazvati samo konstantnu količinu mrežnog saobraćaja. Pošto je emitovanje početnog sažetka ograničeno na tri čvora i konvergencija podataka se dešava kroz prilično konstante ACK i ACK2, neće biti mnogo mrežnog skoka. Mada, ako se čvor podigne, svi čvorovi će možda želeti da pošalju podatke tom razvopravnom uređaju, što će izazvati 'Oluju tračeva'.

Dakle, kako novi čvor dobije ideju o tome s kim da počne da ogovara? Cassandra ima mnogo implementacija dobavljača semena koje daju listu početnih adresa novom čvoru i odmah počinje da ogovara jednu od njih. Nakon svog kruga ogovaranja, sada će posedovati informacije o članstvu u klasteru o svim ostalim čvorovima u klasteru i onda može da ogovara sa ostalima.

Kako se saznaje da li je čvor UP/DOWN? Detektor kvarova je jedna komponenta unutar Cassandre (samo primarna klasa tračeva može pored toga označiti čvor UP) koja to može učiniti. To je slušalac otkućaja srca i obeležava vremenske oznake i čuva zaostale intervale u kojima prima ažuriranja otkućaja srca od svakog vršnjaka. Na osnovu prijavljenih podataka, utvrđuje da li je vršnjak UP/DOWN.

Operacije pisanja ostaju nepromenjene. Ako čvor ne dobije potvrdu za upis ravnopravnom uređaju, on je jednostavno čuva kao nagoveštaj. Čvorovi će prestati da šalju zahteve za čitanje vršnjaku u DOWN stanju i verovatno se može isprobati ogovaranje pošto je neodostupan čvor. Sve sesije

popravke, strimovanja su takođe zatvorene kada je uključen nedostupan čvor.

Cassandra ima još jednu komponentu koja se zove Dinamic Snitch, koja beleži i analizira latencije zahteva za čitanje ka ravnopravnim čvorovima. On rangira latencije vršnjaka u stalnom prozoru i ponovo ga izračunava svakih 100ms i resetuje rezultate svakih 10 minuta da bi omogućilo bilo koji drugi događaj koji odlaže vreme odgovora vršnjaka. Na ovaj način, Dinamic Snitch pomaže da se identifikuju spori čvorovi i da se izbegnu kada se upušta u Gossip. [6]

### **3.6.1. Otkrivanje kvarova i oporavak**

Detekcija kvara je metod za lokalno određivanje na osnovu stanja ogovaranja i istorije da li je drugi čvor u sistemu u kvaru ili se vratio. Cassandra koristi ove informacije da izbegne rutiranje zahteva klijenata ka nedostupnim čvorovima kad god je to moguće.

Proces ogovaranja prati stanje od drugih čvorova i direktno (čvorovi koji ga ogovraju direktno) i indirektno (čvorovi o kojima komuniciraju iz druge ruke, iz treće ruke i tako dalje). Umesto da ima fiksni prag za obeležavanje neispravnih čvorova, Cassandra koristi mehanizam za otkrivanje akrula da izračuna prag po čvoru koji uzima u obzir performanse mreže, radno opterećenje i istorijske uslove. Tokom razmene tračeva, svaki čvor održava klizni prozor između vremena dolaska tračeva sa drugih čvorova u klasteru. Konfigurisanje *phi\_convict\_threshold* svojstvo podešava osetljivost detektora kvarova. Niže vrednosti povećavaju verovatnoću da će čvor koji ne reaguje biti označen kao neaktivan. Koristi se podrazumevana vrednost u većini slučajeva, ali se može povećati na 10 ili 12 za Amazon EC2 (zbog zagušenja mreže). U nestabilnim mrežnim okruženjima (kao što je EC2 ponekad), podizanje vrednosti na 10 ili 12 pomaže u sprečavanju lažnih grešaka. Ne preporučuju se vrednosti veće od 12 i manje od 5.

Kvarovi čvorova mogu biti posledica različitih uzroka kao što su kvarovi na hardveru i prekidi mreže. Prekidi čvorova su često prolazni, ali mogu trajati duže vreme. Pošto ispad čvora retko označava trajni odlazak iz klastera, to ne dovodi automatski do trajnog uklanjanja čvora iz prstena. Drugi čvorovi će povremeno pokušavati da ponovo uspostave kontakt sa neuspešnim čvorovima da vide da li su rezervni. Da bi trajno promenili članstvo čvora u klasteru, administratori moraju eksplicitno da dodaju ili uklone čvorove iz Cassandra klastera koristeći uslužni program nodetool.

Kada se čvor vrati na mrežu nakon prekida rada, možda je propustio zapise za repliku podataka koje održava. Mehanizmi za popravku postoje za oporavak propuštenih podataka, kao što su nagoveštena primopredaja i ručna popravka sa popravkom nodetool-a. Dužina prekida će odrediti koji mehanizam popravke se koristi da bi podaci bili dosledni. [8]

### **3.7. Provera statusa klastera Cassandre**

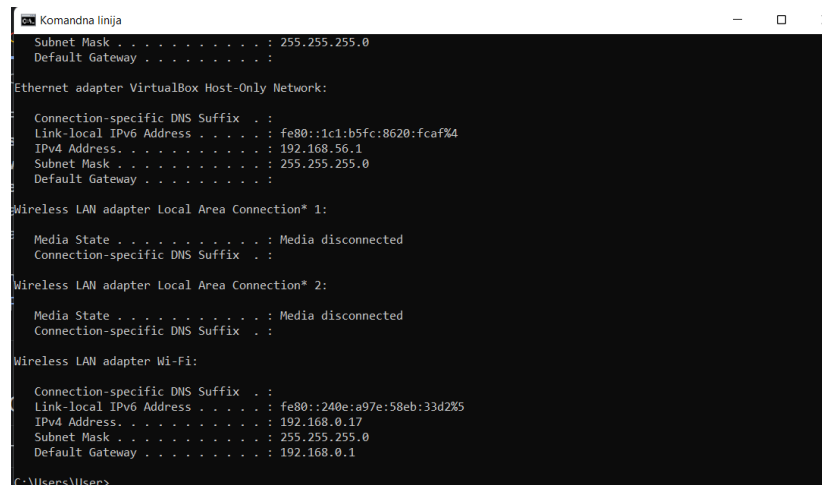
Među najboljim praksama Cassandra klastera je redovna provera zdravlja Cassandra klastera. To se radi uz pomoć Cassandrinog nodetool-a, alata za praćenje koji pomaže u obnavljanju rutinskih zadataka održavanja i nadgledaju Cassandra klastera.

Postoje tri važne naredbe nodetool-a koje se odnose na zdravlje Cassandra klastera kojih

treba biti svestan: status nodetool-a (`nodetool status`), informacije o nodetool-u (`nodetool info`) i `tpstats` nodetool-a (`nodetool tpstats`). Provera zdravlja Cassandre može se izvršiti sa statusom nodetool-a. Komande za status nodetool-a omogućavaju proveru status Cassandra klastera i da se vide stvari kao što su distribucija podataka među čvorovima, da li su čvorovi gore ili dole, stanja čvoroca, učitavanje podataka čvora, brojevi tokena i povezane informacije. Komanda *nodetool info* nudi informacije o čvoru, uključujući status aktivnog ili pasivnog ogovaranja, vreme neprekidnog rada, opterećenje diska, informacije o keš memoriji, vremena početka (generacije), korišćenje memorije u hrpi i još mnogo toga. Komanda *nodetool tpstats* prikazuje statistiku korišćenja pula niti u svakoj fazi.

## 4. Praktična primena

Za potrebe demonstracije rada klastera, izabrala sam da to bude klaster sa dve mašine. Jedna mašina je laptop, a druga klasičan računar. Kako bih podesila Cassandra klaster, koristim dve mašine, a njihove ip adrese su: 192.168.0.17 (laptop) i 192.168.253.1 (računar). Kao proveru ip adrese mašine kucamo u cmd komandu ipconfig i dobijamo adresu (primer je prikazan na slici):



```
Komandna linija
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Ethernet adapter VirtualBox Host-Only Network:

Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::1c1:b5fc:8620:fcaf%4
IPv4 Address. . . . . : 192.168.56.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Wireless LAN adapter Local Area Connection* 1:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 2:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::240e:a97e:58eb:33d2%5
IPv4 Address. . . . . : 192.168.0.17
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.0.1

C:\Users\User>
```

Slika: Ip adresa

Prvo što je bilo potrebno je da se otvore sledeći portovi zaštitnog zida na obe mašine: 7000, 7001, 7199, 9042, 9160 i 9142. Za otvaranje portova zaštitnog zida, neophodno je u kontrolnom panelu otvoriti zaštitni zid Windows-a i nakon toga izabrati opciju u delu sa izuzecima za dodavanje porta. Portu se dodeljuje i izabrano ime, recimo 'CassAppPort7000'.

Ne postoji centralni gospodar u klasteru Cassandra. Umesto toga, potrebno je učiniti da svaki od njih bude svestan drugih i da oni rade zajedno. Prvo se treba urediti /etc/cassandra/cassandra.yaml fajl na obe mašine i postaviti sledeće vrednosti koje su navedene ispod. Za sada se ime klastera ne menja još uvek. To će se učiniti kasnije.

- Seeds-postavlja se IP adresa na jednoj mašini da bude seme. Nije neophodno da sve mašine budu seme. Seme su čvorovi koje Cassandra čvorovi koriste kada se pokrene Cassandra i počinje da pronalazi druge čvorove.
- listen\_address – IP adresa za pokretanje Cassandre.
- Endpoint\_snitch – koristi se za određivanje gde se usmeravaju podaci i šalju replike. Koristi se podrazumevano ispod. Ima ih nekoliko. Ostali su svesni stalka, što znači da ne bi postavili repliku na isti fizički stalak za skladištenje kao drugi. Ako se to uradi i ceo stalak pokvari, podaci bi mogli biti izgubljeni. Postoji čak i jedan dizajniran za Amazon EC2 koji može širiti podatke širom Amazon zona.

Podešavanje mašine 192.168.253.1 (računar):

```
endpoint_snitch: SimpleSnitch
- seeds: "seeds: 192.168.0.17"
listen_address: 192.168.253.1
```

Podešavanje mašine 192.168.0.17 (laptop):

```
endpoint_snitch: SimpleSnitch
- seeds: "seeds: 192.168.0.17"
listen_address: 192.168.0.17
```

Nakon ovih podešavanja, pokrećemo obe mašine sa:

```
sudo service cassandra start
```

Zatim se sačeka nekoliko sekundi za podatke, a zatim se pokreće na obe mašine:

```
nodetool status
```

Trebalo bi da prikaže oba čvora:

```
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address          Load          Tokens         Owns (effective)  Host ID
Rack
UN  192.168.0.17     235.79 KiB    256            100.0%            fb1d89bb-
cbe2-488f-b2e7-da145bd2dde7  rack1
UN  192.168.253.1    176.01 KiB    256            100.0%            472fd4f0-
9bb3-48a3-a933-9c9b07f7a9f6  rack1
```

Ukoliko se dobije bilo kakva poruka o grešci, treba se pogledati u /var/log/cassandra/system.log.

Sada se menja ime podrazumevanog klastera. Pokreće se cqlsh, a zatim se nalepi SQL ispod. Cassandra ne replicira ovu sistemsku promenu u celom klasteru, tako da ovo mora da se pokrene na obe mašine:

```
UPDATE system.local SET cluster_name = 'Cluster2' where key='local';
```

Sada se uređuje /etc/cassandra/cassandra.yaml i menja se ime klastera u šta god se želi. Trebalo bi da bude isto na obe mašine, na primer:

```
cluster_name: 'Cluster2'
```

Zatim:

```
sudo service cassandra restart
```

Ponovo se pokreće ova provera:

```
nodetool status
```

```

Datacenter: datacenter1
=====
Status=Up/Down
|| State=Normal/Leaving/Joining/Moving
-- Address          Load          Tokens      Owns (effective)  Host ID
Rack
UN  192.168.0.17    306.4 KiB    256          100.0%             fb1d89bb-cbe2-
488f-b2e7-da145bd2dde7  rack1
UN  192.168.253.1   294.71 KiB   256          100.0%             472fd4f0-
9bb3-48a3-a933-9c9b07f7a9f6  rack1

```

Sada, napravićemo neke podatke. Videćemo da se podaci uneti na jednom čvoru repliciraju na drugi. Treba nalepiti sledeće SQL u csq1:

```

CREATE KEYSPACE "OnlineMusicConcert"
WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 3
};
CREATE TABLE "Koncert" (
"koncertID" text,
ime text,
opis text,
organizator text,
sponsor text
tip text,
PRIMARY KEY ("koncertID")
);

CREATE TABLE "Zakazivanje" (
"zakID" text,
"izvodjacID" text,
"bendID" text,
"koncertID" text,
vreme text,
datum text,
dodatneinf text,
PRIMARY KEY ("zakID")
);

```



```

INSERT INTO "Koncert" ("koncertID",ime,opis,organizator,sponsor,tip)
VALUES('122','Moderna','Dobrodosli','Grad Nis','Bmw','20-te');
INSERT INTO "Koncert" ("koncertID",ime,opis,organizator,sponsor,tip)
VALUES('133','UvekZabava','Dobar provod','Firma Anoo','Stark','Pop');
INSERT INTO "Koncert" ("koncertID",ime,opis,organizator,sponsor,tip)
VALUES('144','Nikad bolje','Dobrodosli!','Srednjoskolci','Restoran
IN','Razno');

```

Zatim se prijavimo na suprotnu mašinu i proveravamo da li su podaci tamo kopirani:

```
select * from koncert;
```

koncertID	ime	opis	organizator	sponsor	tip
122	Moderna	Dobrodosli	Grad Nis	Bmw	20-te
133	UvekZabava	Dobar provod	Firma Ano	Stark	Pop
144	Nikad bolje	Dobrodosli!	Srednjoskolci	Restoran In	Razno

## 5. Zaključak

Grupisanje baza podataka je proces kombinovanja više od jednog servera ili instanci koje povezuju jednu bazu podataka. Ponekad jedan server možda nije adekvatan za upravljanje količinom podataka ili brojem zahteva, tada je potreban klaster podataka. Glavni razlozi za klasterisanje baze podataka su prednosti koje server dobija, redudantnost podataka, balansiranje opterećenja, visoka dostupnost, i na kraju, nadzor i automatizacija.

Klaster kod Cassandra baze podataka sastoji se od nekoliko centara podataka, gde svaki od centra podataka sadrži čvorove, koji podrazumevano se sastoje od 256 virtualnih čvorova. Virtualni čvorovi pružaju veću fleksibilnost sistemu.

Korišćenje Cassandra klastera omogućuje u zavisnosti od potrebe inicijalizaciju klastera sa sa više čvorova u jednom centru podataka ili više centara podataka. Procedura dodavanja novih čvorova u klaster obuhvata niz koraka koji se treba primeniti, kako bi dodavanje i upotreba bila uspešna. Ukoliko neki od čvorova nije potreban više za korišćenje (aktivan je) ili ukoliko je došlo do kvara nekog od čvorova, moguće je zameniti ga. Brisanje čvora ukoliko iz nekog razloga je to neophodno ili je došlo do kvara takođe se može izvršiti.

Kako bi čvorovi u Cassandra klasteru komunicirali koristi se Gossip protokol. Gossip je efikasan, lagan, pouzdan inter-nodalni protokol emitovanja za širenje podataka. Decentralizovan je 'epidemijski', tolerantan na greške i protokol za međusobnu komunikaciju. Cassandra koristi ogovaranje za otkrivanje kolega i širenje metapodataka.

Proces ogovaranja prati stanje od drugih čvorova i direktno (čvorovi koji ga ogovaraju direktno) i indirektno (čvorovi o kojima komuniciraju iz druge ruke, iz treće ruke i tako dalje) i na osnovu toga može detektovati ukoliko je došlo do nekog kvara ili greške. Kvarovi čvorova mogu biti posledica različitih uzroka kao što su kvarovi na hardveru i prekidi mreže. Prekidi čvorova su često prolazni, ali mogu trajati duže vreme. Pošto ispad čvora retko označava trajni odlazak iz klastera, to ne dovodi automatski do trajnog uklanjanja čvora iz prstena. Drugi čvorovi će povremeno pokušavati da ponovo uspostave kontakt sa neuspelim čvorovima da vide da li su rezervni. Da bi trajno promenili članstvo čvora u klasteru, administratori moraju eksplicitno da dodaju ili uklone čvorove iz Cassandra klastera koristeći uslužni program.

Na kraju, na osnovu teorijskog dela i praktičnog dela primene korišćenja klastera kod Cassandra baze podataka, može se reći da je neophodno dobro poznavati funkcionisanje Cassandra baze podataka, njenih osnovnih koncepata kako bi se mogli koristiti klasteri ove baze podataka. Inicijalizacija klastera obuhvata niz koraka kojih se treba pridržavati i prethodno dobro proučiti literaturu. Svakako, korišćenje klastera omogućava lakši i brži pristup bazi podataka, gde sa više mesta joj se može pristupati i koristiti.

## 6. Literatura

- [1] What is Database Clustering – Introduction and brief explanation, dostupno na: <https://www.ndimensionz.com/2018/01/05/what-is-database-clustering-introduction-and-brief-explanation/>
- [2] Cluster, Datacenters, Racks and Nodes in Cassandra, dostupno na: <https://www.baeldung.com/cassandra-cluster-datacenters-racks-nodes>
- [3] Initializing a multiple node cluster (single datacenter), dostupno na: <https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/initialize/initSingleDS.html>
- [4] What is Cassandra Cluster & Cluster Builder, dostupno na: <https://data-flair.training/blogs/cassandra-cluster/>
- [5] Cassandra Cluster, dostupno na: <https://www.scylladb.com/glossary/cassandra-cluster/>
- [6] The Gossip Protocol – Inside Apache Cassandra, dostupno na: <https://www.linkedin.com/pulse/gossip-protocol-inside-apache-cassandra-soham-saha/>
- [7] Adding nodes to an existing cluster dostupno na: <https://docs.datastax.com/en/archived/cassandra/3.0/cassandra/operations/opsAddNodeToCluster.html>
- [8] Failure detection and recovery, dostupno na: <https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/architecture/archDataDistributeFailDetect.html>
- [9] Operations, dostupno na: <https://docs.datastax.com/en/archived/cassandra/3.0/cassandra/operations/operationsTOC.html>