|  |  |  |
| --- | --- | --- |
| **Unigrb** | UNIVERZITET U NIŠU  ELEKTRONSKI  FAKULTET | **logo_1960_4** |

SISTEMI ZA UPRAVLJANJE

BAZAMA PODATAKA

SEMINARSKI RAD

**REPLIKACIJA KOD**

**APACHE CASSANDRA**

**BAZE PODATAKA**

|  |  |
| --- | --- |
| Mentor:  Aleksandar Stanimirović | Student:  Dragana Korunović  1220 |

Niš, maj 2021. godine

**Sadržaj**

[1. Uvod 2](#_Toc73520426)

[2. Arhitektura 3](#_Toc73520427)

[2.1. Zahtevi arhitekture 3](#_Toc73520428)

[2.2. Karakteristike arhitekture 3](#_Toc73520429)

[2.3. Komponente arhitekture 4](#_Toc73520430)

[2.3.1. Rack 4](#_Toc73520431)

[2.3.2. Mrežna topologija 5](#_Toc73520432)

[2.3.3. Snitch-evi 5](#_Toc73520433)

[2.3.4. Virtuelni čvorovi 5](#_Toc73520434)

[2.4. Particionisanje podataka 6](#_Toc73520435)

[2.5. Gossip protokol 7](#_Toc73520436)

[2.5.1. Seed čvorovi 8](#_Toc73520437)

[3. Replikacija podataka 8](#_Toc73520438)

[3.1. Faktor replikacije 9](#_Toc73520439)

[3.2. Strategija replikacije 9](#_Toc73520440)

[3.2.1. Jednostavna strategija 9](#_Toc73520441)

[3.2.2. Strategija mrežne topologije 9](#_Toc73520442)

[3.3. Kreiranje klastera 10](#_Toc73520443)

[3.4. Nivo konzistencije 13](#_Toc73520444)

[3.4.1. Nivo konzistencije upisa 13](#_Toc73520445)

[3.4.2. Nivo konzistencije čitanja 15](#_Toc73520446)

[3.4.3. O nivoima QUORUM i LOCAL\_QUORUM 16](#_Toc73520447)

[3.5. Upis i čitanje u Cassandra čvorovima 17](#_Toc73520448)

[3.5.1. Operacija upisa 17](#_Toc73520449)

[3.5.2. Operacija čitanja 19](#_Toc73520450)

[4. Zaključak 22](#_Toc73520451)

[5. Literatura 23](#_Toc73520452)

# Uvod

Jedan od sistema za upravljanje bazama podataka koji uspešno radi sa big data[[1]](#footnote-1) je Apache Cassandra. To je nerelaciona baza podataka – ne poseduje strogo definisanu šemu podataka. Ova baza podataka obezbeđuje skalabilnost na nivou stotina hiljada čvorova, ne poseduje jednu tačku prekida i distribuirane je arhitekture.

Neki od elemenata arhitekture Cassandra-e su čvor, rack, centar podataka i klaster. Podaci se lako i transparentno raspoređuju u navedenim elementima arhitekture, od kojih je osnovni element čvor. Način raspoređivanja određuje particioner – hash vrednost particionog ključa vrste podataka[[2]](#footnote-2).

Pošto se otkaz sistema može javiti zbog hardverskih i mrežnih problema, uvedena je redundantnost kroz replikaciju podataka. Replikacija podrazumeva dva faktora – faktor replikacije i strategiju replikacije. Prvi određuje ukupan broj kopija podataka smeštenih na različitim čvorovima sistema, a drugi njihove lokacije. Još jedan značajan faktor je nivo konzistencije, koji se odnosi na ažurnost i sinhronizovanost svih kopija neke vrste i utiče na upis i čitanje podataka.

U radu će biti obrađeni pomenuti pojmovi, sa akcentom na replikaciji podataka, uz odgovarajuće primere. Naredno poglavlje posvećeno je arhitekturi sistema – njenim zahtevima, karakteristikama, komponentama, kao i particionisanju podataka i komunikaciji između čvorova. U trećem poglavlju biće reči o replikaciji podataka, njenim faktorima, o podešavanju nivoa konzistencije, čitanju i upisu podataka. Uz to je dat primer kreiranog klastera sa unetim podacima i izvršenim upisom i čitanjem. Poslednje poglavlje predstavlja zaključak rada.

# Arhitektura

Cassandra je dizajnirana kao peer-to-peer sistem koji kreira i rapoređuje kopije podataka u grupama čvorova. Podaci su organizovani u tabele (familije kolona), gde svaka vrsta tabele poseduje particioni ključ. Hash funkcijom definisanom za svaki od klastera i primenjenom nad particionim ključem se određuje kako će podaci biti distribuirani. Najpre će biti sagledani zahtevi koje arhitektura Cassandra-e treba da ispunjava.

## Zahtevi arhitekture

Najbitniji zahtev arhitekture Cassandra-e je da obezbedi da ne postoji jedna tačka prekida, što znači da bi sistem prestao da funkcioniše ako jedan od čvorova otkaže. Drugi zahtev je da poseduje skalabilnost takvu da klaster može da sadrži stotine hiljada čvorova, bez prekida sa radom pri dodavanju novog. Takođe, arhitektura treba da bude distribuirana, tako da i obrada i podaci budu distribuirani. Da bi sistem mogao da se koristi u realnom vremenu, potrebni su brz upis i čitanje.

## Karakteristike arhitekture

Neke od karakteristika arhitekture Cassandra-e su sledeće:

* Ne poseduje master i slave čvorove.
* Arhitektura je prstenastog tipa, tj. čvorovi su logički smešteni u prsten.
* Podaci se repliciraju u čvorovima radi posizanja redundantnosti, a nju određuje faktor replikacije, definisan u zavisnosti od važnosti podataka.
* Podaci se čuvaju u memoriji i smeštaju na disk u određenim trenucima.
* Hash vrednosti ključeva koriste se za distribuciju podataka u čvorovima klastera.
* Podržano više centara podataka, po kojima se vrši replikacija. Mogu se čuvati tri kopije u jednom centru podataka i četvrta u udaljenom centru, koji predstavlja backup. Podaci koji se čitaju se najpre traže u lokalnom centru podataka.
* Distribucija podataka između čvorova je transparentna, što znači da se na osnovu podatka može utvrditi njegova lokacija u klasteru.
* Pri čitanju podataka svaki čvor može da prihvati zahtev i obezbedi ih, ako ih poseduje, ili prosledi zahtev čvoru koji poseduje tražene podatke.
* Nivo konzistencije može da se definiše, gde njegovo povećanje negativno utiče na performanse.
* Transakcije su trajne jer se upisuju u commitlog na disku.
* Razmena podataka između čvorova se odvija putem Gossip protokola.

Pomenute komponente biće objašnjene u narednom poglavlju.

## Komponente arhitekture

Arhitekura Cassandra-e sastoji se od sledećih komponenata (Slika 1):

* ***Čvor* – o**snovna komponenta arhitekture u kojoj se skladište podaci.
* ***Rack*** – grupa čvorova (mašina) koja je smeštena na jednom mestu.
* ***Centar podataka*** – kolekcija više rack-ova ili čvorova.
* ***Klaster* –** kolekcija više centara podataka, čiju mrežnu topologiju definiše *snitch*.
* ***Commit Log –*** mesto gde se upisuje svaka operacija upisa; služi za oporavak od otkaza.
* ***Memtable –* nakon upisa u** Commit log, podaci se privremeno upisuju u Memtable.
* ***SSTable –* nakon prelaska određene granice u okviru memtable-a, podaci se prebacuju na ovu datoteku na disku. Ona se koristi za ažuriranje odgovarajuće tabele.**

U nastavku će biti obrađeni najznačajniji elementi arhitekture Cassandra-e.

|  |  |
| --- | --- |
| [Cassandra Architecture & Replication Factor Strategy Tutorial](https://www.guru99.com/images/cassandra/021116_0524_CassandraAr1.png) |  |
| Slika – čvorovi, klaster i centri podataka | Slika – centar podataka sa svojim rack-ovima i čvorovima |

### Rack

Rack je grupa mašina koja je fizički smeštena na jednom mestu. Svaka od mašina (čvorova) poseduje svoj procesor, memoriju i disk, ali to ne važi i za rack u celini (Slika 2). Neke od osobina rack-a su sledeće:

* Svaka od mašina u rack-u je povezana na mrežni switch tog rack-a.
* Mrežni switch rack-a povezan je sa klasterom i njegov kvar može prouzrokovati pad rack-a.
* Mašine u rack-u imaju zajednički izvor struje, zbog koga on takođe može otkazati.
* U slučaju da rack otkaže, nijednoj njegovoj mašini se ne može pristupiti, tj. svaki od čvorova nije u funkciji.

### Mrežna topologija

Mrežna topologija se odnosi na organizaciju čvorova, rack-ova i centara podataka. Mrežna topologija klastera može se odrediti preko datoteke cassandra-topology.properties. Za svaki čvor u klasteru se na osnovu IP adrese ili hostname-a može specificirati kom centru podataka i rack-u pripada, kao i odrediti podrazumevana lokacija čvora, u slučaju da to nije specificirano u datoteci. Primer izgleda datoteke cassandra-topology.properties dat je na slici 3. Redom su navedene IP adrese čvorova i njihove lokacije, kao i podrazumevana lokacija nesvrstanih čvorova.

### Snitch-evi

Cassandra koristi snitch da bi se definisalo kako su čvorovi grupisani u mrežnoj topologiji klastera (grupišu se po rack-ovima i centrima podataka), u datoteci cassandra.yaml. Omogućava prevođenje IP adrese čvora u centar podataka i rack kome pripada. Najznačajnije vrste snitch-eva su sledeće:

1. *Jednostavni snitch (Simple Snitch)* – podrazumevana vrsta snitch-a, koja se koristi u jednostavnoj strategiji replikacije, za usamljene centre podataka bez rack-ova.
2. *GossipingPropertyFileSnitch –* preporučen za upotrebu u produkciji. U datoteci cassandra-topologies.properties se pronalaze centar podataka i rack kojima čvor pripada, kao i u datoteci cassandra-rackdc.properties.
3. *Snitch koji zaključuje na osnovu rack-a (Rack Inferring Snitch)* – zaključuje o topologiji mreže analizom IP adrese čvorova. On pretpostavlja da drugi oktet identifikuje centar podataka gde se čvor nalazi, a treći njegov rack.
4. *Snitch datoteke sa osobinama (Property File Snitch)* – određuje lokacije čvorova posmatrajući detalje o mreži koji je korisnik definisao u datoteci cassandra-topology.properties. Koristi se kada se kreira više centara podataka sa više rack-ova.
5. *EC2 Snitch* – koristi se samo kod razvoja na Amazon EC2 web servisu. Ovaj snitch ne koristi IP adresu da otkrije lokaciju čvora, već AWS API, preko koga zahteva podatke o regionu i oblasti dostupnosti.

### Virtuelni čvorovi

Virtuelni čvorovi u Cassandra klasteru (vnodes) mogu da se definišu za svaki od fizičkih čvorova u njemu. Fizički čvor u prstenu podrazumevano sadrži 256 virtuelnih. Virtuelni čvorovi doprinose granularnosti pri particionisanju podataka, gde se oni smeštaju u zavisnosti od hash vrednosti ključa. Pri dodavanju novog čvora, virtuelni čvorovi na njemu dobijaju jednake delove postojećih podataka. Na slici 4 dat je primer klastera sa četiri fizička čvora, od kojih svaki poseduje četiri virtuelna, pa ih je ukupno 16. Tada, ako treba smestiti 64 GB podataka u klasteru, svaki od virtuelnih čvorova će sadržati po 4 GB podataka. Ako se ubaci novi čvor nalik postojećim, po 3.2 GB podataka će biti u svakom od virtuelnih čvorova.

|  |  |
| --- | --- |
|  |  |
| Slika 3 – datoteka cassandra-topology.properties | Slika 4 – klaster sa četiri fizička čvora |

## Particionisanje podataka

Cassandra je distribuirani sistem za upravljanje bazama podataka, koji lako i transparentno raspoređuje podatke u svim čvorovima klastera. Svaki čvor je odgovoran za određeni deo baze podataka. Podaci se ubacuju u tabelu i dodeljuje im se ključ vrste, koji ih jedinstveno identifikuje. Da bi se izvršila raspodela podataka, koristi se particioner. To je funkcija koja računa token – hash vrednost particionog ključa vrste, po kojoj se svaka vrsta raspoređuje u klasteru. Pri tome je potrebno odrediti vrstu particionera preko opcije partitioner, u datoteci za konfiguraciju (cassandra.yaml). Nakon što se klaster inicijalizuje, ova opcija se ne može ažurirati bez ponovnog učitavanja svih njegovih podataka. Mogu se izdvojiti tri vrste particionera:

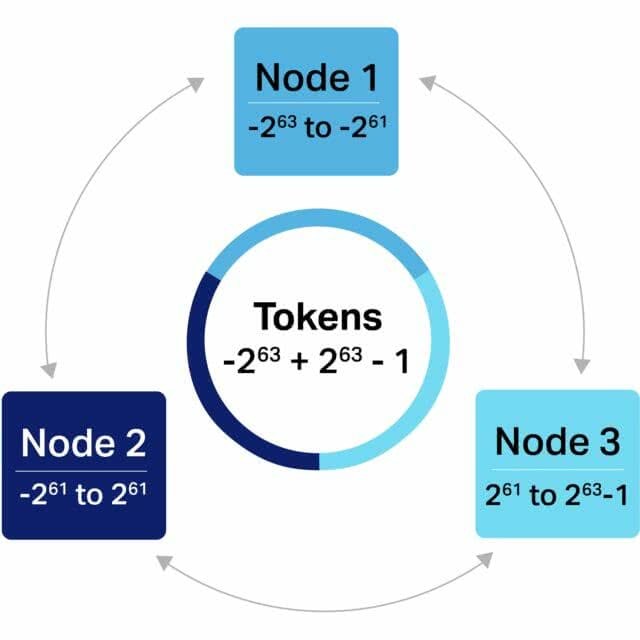
* *Murmur3Partitioner* (podrazumevani) – uniformno raspoređuje podatke u klasteru na osnovu MurmurHash hash vrednosti. Dobar je izbor za skoro svaki novokreirani klaster.
* *RandomPartitioner* – uniformno raspoređuje podatke u klasteru na osnovu MD5 hash vrednosti.
* *ByteOrderedPartitioner* – čuva leksički uređene distribuirane podatke na osnovu bajtova ključa.

Murmur3Partitioner i RandomPartitioner koriste tokene za uniformnu raspodelu podataka iz svih tabela u svakom od čvorova. ByteOrderedPartitioner se koristi za uređeno particionisanje, što dozvoljava da se vrši pretraga podataka uređenih na osnovu particionog ključa po njegovoj vrednosti. Ovo je pogodno za pretragu podataka u određenom opsegu vrednosti ključa. Međutim, nije preporučljivo koristiti ByteOrderedPartitioner jer je tada teško balansirati opterećenje i sekvencijalni upisi bi mogli da budu neefikasni.

Particionisanje podataka je horizontalno i vrši se na sledeći način:

* Računa se hash vrednost na osnovu primarnog ključa podatka (token).
* Hash vrednost se mapira na čvor u klasteru.
* Prva kopija podatka se čuva u trenutnom čvoru.
* Distribucija je transparentna jer može da se izračuna hash vrednost i odredi gde će određena vrsta biti smeštena.

Veličina particije treba da bude manja od 100 MB, a idealno iznosi manje od 10 MB. Na slici 5 može se videti kako bi podaci bili raspoređeni u tri čvora, u zavisnosti od hash vrednosti particionog ključa. U ovom primeru se radi jednostavnosti posmatraju čvorovi, dok se raspoređivanje zapravo vrši po virtuelnim čvorovima.



Slika – particionisanje podataka u zavisnosti od vrednosti tokena

## Gossip protokol

Cassandra koristi gossip protokol za komunikaciju između čvorova u klasteru. Ovaj protokol otkriva lokaciju ostalih čvorova i podatke o njihovom stanju. Pokreće se periodično na svakom čvoru i razmenjuje podatke o svom stanju sa tri druga čvora u klasteru. Na taj način se informacije propagiraju za par sekundi, u slučaju postojanja nekoliko hiljada čvorova.

Gossip protokol se odvija na sledeći način: najpre se čvor povezuje sa tri druga čvora (korak 1). U koraku 2 se svaki od ta tri čvora povezuje sa još tri druga čvora. To ukupno čini 13 čvorova (Slika 6). Ovaj protokol koristi seed čvorove, koji su opisani u nastavku.

|  |  |
| --- | --- |
|  |  |
| Slika 6 – ilustracija gossip protokola | Slika 7 – ilustracija rada seed čvorova |

### Seed čvorovi

Seed čvorovi se koriste za pokretanje gossip protokola, kada je čvor pokrenut ili ponovo pokrenut. Oni se specificiraju u datoteci cassandra.yaml. Preko ovih čvorova se postiže stabilno stanje, u kom je svaki čvor povezan sa ostalima.

Na slici 7 može se videti primer pokretanja klastera koji se sastoji od četiri čvora, od kojih su dva seed tipa. Pri pokretanju se čvorovi koji nisu seed čvorovi povezuju sa seed čvorovima, kao i seed čvorovi međusobno. Sada svaki čvor zna za postojanje drugih i postignuto je stabilno stanje.

# Replikacija podataka

Uzrok otkaza u sistemu može da bude problem sa hardverskim komponentama ili mrežnom konekcijom. Zbog toga, da bi se osigurala otpornost na otkaze i nepostojanje jedne tačke prekida, uvodi se replikacija podataka. To znači da svaka vrsta poseduje svoje kopije i da se svaka smešta u čvorove koji su deo klastera. Postoje dva faktora za raspoređivanje kopija podataka:

* *Faktor replikacije* – određuje ukupan broj replika smeštenih na različitim čvorovima.
* *Strategija* *replikacije* – određuje gde će kopije biti smeštene.

## Faktor replikacije

Ukupan broj replika nekog keyspace-a u klasteru predstavlja faktor replikacije, gde je svaka replika od iste važnosti. Da ne bi postojala jedna tačka prekida, faktor replikacije mora iznositi barem tri, što je i njegova podrazumevana vrednost za svaki centar podataka u sistemu sa tri ili više Cassandra čvorova. U slučaju veoma značajnih podataka ovaj faktor iznosi 5. Iako faktor replikacije ne treba da premašuje broj čvorova u klasteru, on se može najpre povećati, a onda dodati željeni broj čvorova.

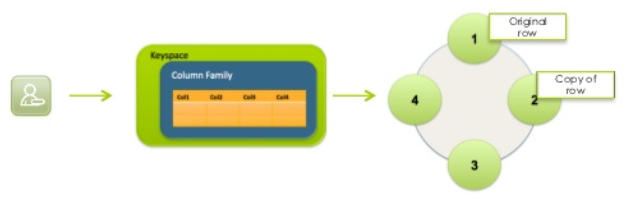
Pri ubacivanju dodatnih Cassandra čvorova u klaster, ne utiče se na podrazumevani faktor replikacije. Na primer, posmatrajmo slučaj da se broj čvorova poveća sa tri na šest, a faktor replikacije iznosi tri. Tada nema dovoljnog broja kopija podataka da se osigura da se pri otkazu određenih čvorova oni ne izgube. Zato faktor replikacije treba da se poveća, što povlači veće kašnjenje pri upisu.

## Strategija replikacije

Strategija replikacije određuje čvorove u kojima su kopije podataka smeštene. U Cassandra-i je moguće definisati replikaciju na nivou čvorova, rack-ova i centara podataka. Postoje dve strategije za replikaciju – jednostavna i strategija mrežne topologije.

### Jednostavna strategija

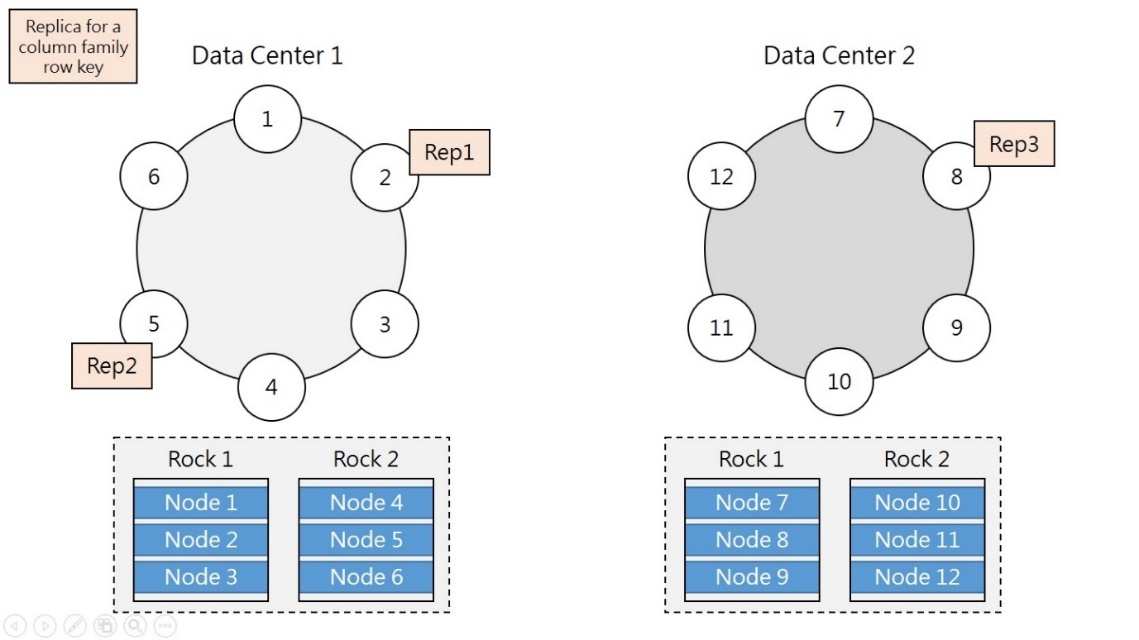
Jednostavna strategija replikacije se koristi u slučaju da postoje jedan centar podataka i rack. Ova strategija podrazumeva da se prva replika smešta na čvor određen particionerom, a ostale na naredne čvorove u prstenu, u smeru kazaljke na satu. Pri tome se ne uzima u obzir topologija, tj. lokacija rack-a ili centra podataka (Slika 8).



Slika – centar podataka sa replikama vrste smeštenim jednostavnom strategijom

### Strategija mrežne topologije

Strategija preporučena u većini primena je strategija mrežne topologije, jer omogućava lakše dodavanje novih centara podataka u bazu. Formalno se koristi čim postoji više od dva centra podataka. Pod ovom strategijom podrazumeva se da se navede željeni broj replika u svakom od centara podataka. Takođe, dozvoljena je replikacija u okviru rack-ova jednog centra podataka ili između više centara podataka. Zbog toga korisnik ima bolju kontrolu nad raspoređivanjem kopija vrsta.



Slika – klaster sa replikama podataka smeštenim strategijom mrežne topologije

Strategija mrežne topologije podrazumeva da se original vrste raspoređuje u skladu sa particionerom; ostale kopije vrste se raspoređuju u okviru istog centra podataka tako što se u smeru kazaljke na satu traži čvor koji je u drugačijem rack-u od onog gde je prethodna kopija. Ako ne postoji takav čvor, replike će biti postavljene u isti rack. Ovakvo raspoređivanje povećava otpornost na otkaze, jer čvorovi u istom rack-u ili istoj fizičkoj grupaciji često otkazuju u isto vreme zbog izvora energije, hlađenja ili smetnji na mreži. Takođe, zbog replikacije podataka između 1-n centara podataka, pri kreiranju keyspace-a se definiše grupa replika i mapira na svaki logički ili fizički centar podataka. Slika 9 prikazuje mogući raspored čvorova u prstenu i njihovo grupisanje u rack-ove i centre podataka. Svaka od tri kopije podatka smeštena je u čvor narednog rack-a, u skladu sa strategijom mrežne topologije.

## Kreiranje klastera

U nastavku će biti prikazan primer kreiranja klastera koji se zasniva na pomenutim pojmovima. Najpre su kreirane dve virtuelne mašine u okruženju Oracle VM VirtualBox i na njima instaliran operativni sistem Ubuntu 20. Svakoj je dodeljen Host-Only Ethernet adapter, sa adresom DHCP servera 192.168.117.2, pa su redom dobile IP adrese 192.168.117.4 i 192.168.117.5. Zatim su otvoreni portovi za TCP komunikaciju koje Cassandra zahteva (Slika 10). U datotekama cassandra.yaml ovih mašina definisane su osobine klastera. To je glavna konfiguraciona datoteka u Cassandra-i, koja sadrži ime klastera, seed čvorove za trenutni čvor, informacije o datoteci sa topologijom i lokaciju datoteke sa podacima. Najvažniji delovi pomenute datoteke su sledeći:

* Mašina 1:

cluster\_name: 'MyCluster'

partitioner: org.apache.cassandra.dht.Murmur3Partitioner

num\_tokens: 256

seed\_provider:

- class\_name: org.apache.cassandra.locator.SimpleSeedProvider

parameters:

- seeds: „192.168.117.4, 192.168.117.5“

listen\_address: 192.168.117.4

rpc\_address: 192.168.117.4

endpoint\_snitch: GossipingPropertyFileSnitch

* Mašina 2:

cluster\_name: 'MyCluster'

partitioner: org.apache.cassandra.dht.Murmur3Partitioner

num\_tokens: 256

seed\_provider:

- class\_name: org.apache.cassandra.locator.SimpleSeedProvider

parameters:

- seeds: „192.168.117.4, 192.168.117.5“

listen\_address: 192.168.117.5

rpc\_address: 192.168.117.5

endpoint\_snitch: GossipingPropertyFileSnitch

Može se videti da je ime klastera promenjeno na MyCluster, koristi se Murmur3Partitioner za particionisanje podataka, broj virtuelnih čvorova po fizičkom čvoru je 256. Oba čvora u klasteru su definisana kao seed čvorovi (navedene su njihove IP adrese) i adrese listen\_address i rpc\_address su postavljene na IP adresu odgovarajuće mašine. Prva određuje adresu preko koje se čvor povezuje sa drugim Cassandra čvorovima, a druga predstavlja adresu preko koje se vrši komunikacija sa klijentskim procesima. Snitch koji se koristi je GossipingPropertyFileSnitch, koji definiše da će se topologija klastera pronaći u datoteci cassandra-topology.properties, a ako ona ne postoji, u datoteci cassandra-rackdc.properties. S obzirom da je datoteka cassandra-topology.properties namerno izbrisana, podaci o centru podataka i rack-u čvora preuzimaju se iz datoteke cassandra-rackdc.properties. Oni su definisani za prvi čvor kao:

dc=dc1

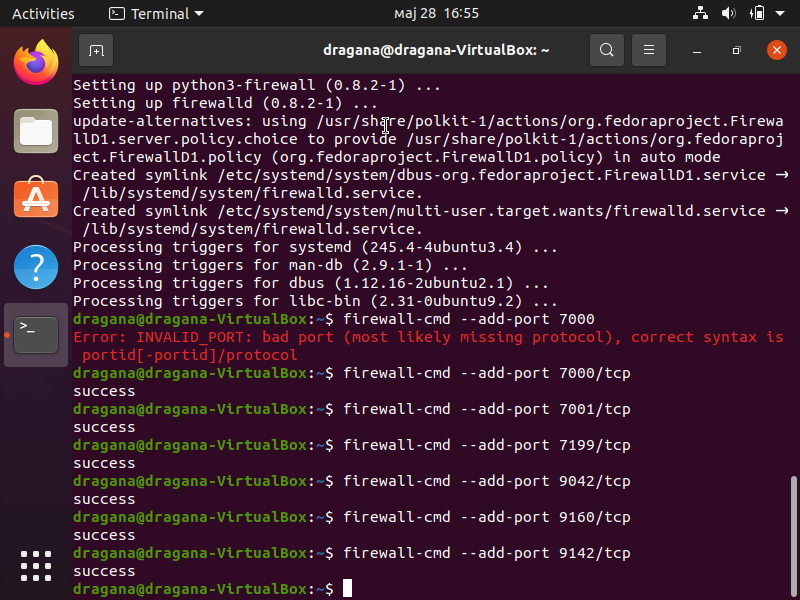
rack=rack1,

a za drugi kao:

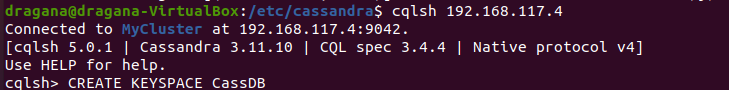
dc=dc1

rack=rack2.

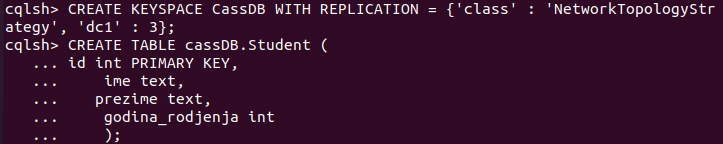
Ovo znači da su oba čvora smeštena u jedan centar podataka – dc1, prvi u rack1, drugi u rack2. Za povezivanje sa klasterom preko terminala se koristi naredba cqlsh[[3]](#footnote-3), uz navođenje IP adrese čvora (Slika 11). Nakon toga je kreiran keyspace cassDB, koji koristi strategiju replikacije NetworkTopologyStrategy. Zato se moraju navesti i centri podataka sa svojim faktorima replikacije, što je u datom primeru dc1, sa faktorom replikacije 3 (Slika 12). Pošto postoje dva čvora, biće kreirane samo dve kopije podataka. Zatim su ubačeni probni podaci, u istom čvoru (Slika 13). Pri pristupu podacima, sa druge virtuelne mašine je pristupljeno njenoj IP adresi preko porta 9042 (klijentski port Cassandra-e) i podacima na drugom čvoru. Na taj način se može videti da su podaci uneti u jednom od čvorova replicirani i na drugom (Slika 14).



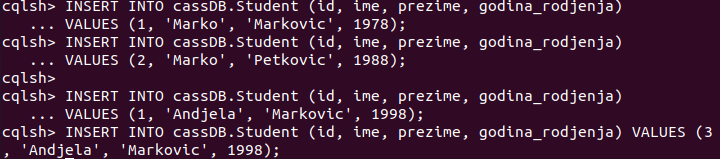
Slika – otvaranje portova za TCP komunikaciju



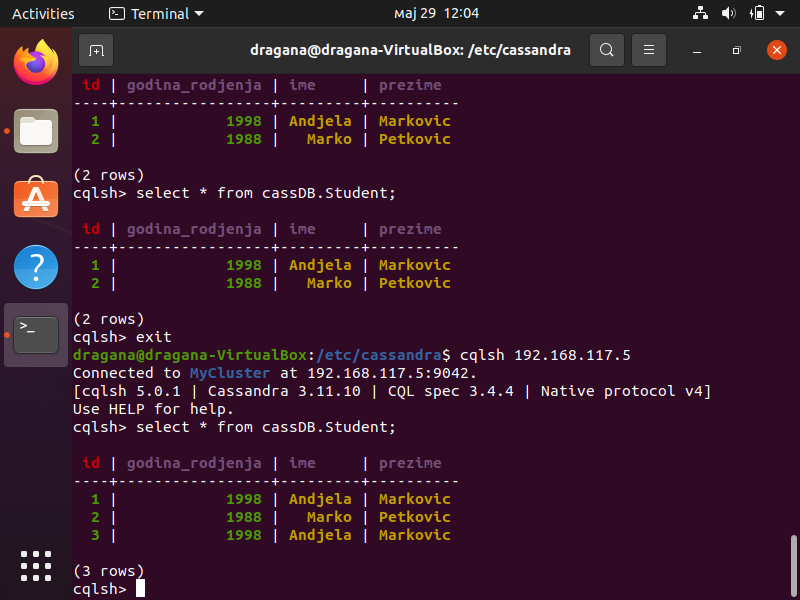
Slika – prijavljivanje na čvor u klasteru, koristeći cqlsh



Slika – kreiranje keyspace-a CassDB i tabele Student



Slika – ubacivanje probnih podataka



Slika – pristup podacima ubačenim sa drugog čvora

## Nivo konzistencije

Konzistencija se odnosi na ažurnost i sinhronizovanost svih kopija jedne vrste u Cassandra-i. Omogućena je podesiva konzistencija za svaku operaciju čitanja i upisa, koju određuje klijentska aplikacija. Nivo konzistencije Cassandra-e je definisan kao minimalni broj Cassandra čvorova koji moraju da prihvate operaciju upisa ili čitanja, pre nego što operacija može da se nazove uspešnom. U nastavku će biti predstavljeni nivoi konzistencije za operacije upisa i čitanja.

### Nivo konzistencije upisa

|  |  |  |
| --- | --- | --- |
| Nivo | Opis | Primena |
| ANY | Upis se mora izvršiti u barem jednom čvoru. Ako su svi čvorovi za upis kopija vrste nefunkcionalni, upis može uspeti nakon [hinted handoff](http://www.datastax.com/documentation/cassandra/2.0/cassandra/dml/dml_about_hh_c.html#concept_ds_ifg_jqx_zj)[[4]](#footnote-4) upisa. U slučaju da nijedan čvor nije u funkciji, podaci za upis nisu čitljivi dok se čvorovi namenjeni kopiji te vrste ne oporave od otkaza. | Omogućava malo kašnjenje i garantuje da upis uvek uspeva. Takođe, obezbeđuje najveću konzistenciju i najmanju dostupnost od svih nivoa konzistencije. |
| ALL | Upis mora biti smešten u commit log i memtable na svim čvorovima u klasteru koji su namenjeni kopijama odgovarajuće vrste. | Omogućava najveću konzistenciju i najmanju dostupnost u odnosu na druge nivoe konzistencije. |
| EACH\_QUORUM | Upis mora da se smesti u commit log i memtable pod kvorumom čvorova za upis kopija u svim centrima podataka. | Koristi se u više centara podataka klastera, da obezbedi konzistenciju na istom nivou u svakom centru. Na primer, u slučaju da je potrebno da čitanje ne uspe kada centar podataka nije u funkciji i QUORUM na njemu ne može biti postignut, bira se ovaj nivo konzistencije. |
| QUORUM | Upis mora biti smešten u commitlog i memtable na osnovu kvoruma replika u svim centrima podataka. | Koristi se u jednom ili više centara podataka da bi se održala jaka konzistecija u klasteru. Koristi se kada može da se toleriše delimičan otkaz. |
| LOCAL\_ONE | Upis mora biti poslat i uspešno potvrđen od strane najmanje jednog čvora u koji se vrsta treba kopirati u lokalnom centru podataka. | Pošto je u klasterima sa više centara podataka nivo konzistencije ONE poželjan, ali ne i saobraćaj između njih, LOCAL\_ONE obezbeđuje da ovo bude slučaj. Može se koristiti u offline centru podataka, da bi se sprečilo automatsko povezivanje sa online čvorovima u drugim centrima podataka, ako taj čvor otkaže. |
| LOCAL\_QUORUM | Upis se mora zapisati u commit log i memtable na osnovu kvoruma čvorova za kopije koji se nalaze u istom centru podataka kao koordinatorski čvor. Ovaj nivo konzistencije izbegava kašnjenje pri komunikaciji između centara podataka. | Koristi se u više centara podataka klastera, uz strategiju postavljanja kopija koja poseduje uvid u rack-ove (strategija mrežne topologije) i pravilno konfigurisanim snitch-em. Ne može da se koristi uz jednostavnu strategiju replikacije. Preporučuje se za održavanje konzistencije na lokalnom nivou (u okviru centra podataka). |
| LOCAL\_SERIAL | Upis mora biti uslovno smešten u commit log i memtable na osnovu kvoruma čvorova za smeštanje kopija u istom centru podataka. | Koristi se za ostvarivanje serijske konzistencije[[5]](#footnote-5) za lightweight tranksakcije[[6]](#footnote-6) tako što se sprečavaju ažuriranja bez zadatog uslova. |
| SERIAL | Upis mora biti uslovno smešten u commit log i memtable na osnovu kvoruma čvorova za smeštanje kopija. | Ista kao za LOCAL\_SERIAL. |
| ONE | Upis mora da se smesti u commit log i memtable barem jednog od čvorova namenjenih kopijama. | Zadovoljava potrebe većine korisnika jer zahtevi za konzistencijom nisu strogi. Čvor za kopiju koji je najbliži koordinatorskom čvoru koji je primio zahvev opslužuje taj zahtev. |
| TWO | Upis mora da se smesti u commit log i memtable barem dva čvora namenjena kopijama. | Sličan nivou ONE. |
| THREE | Upis mora da se smesti u commit log i memtable barem tri čvora namenjena kopijama. | Sličan nivou TWO. |

### Nivo konzistencije čitanja

|  |  |  |
| --- | --- | --- |
| Nivo | Opis | Primena |
| ALL | Vraća vrstu sa najnovijom vremenskom oznakom nakon odgovora svih kopija. Operacija čitanja neće uspeti ako nema odgovora kopije. | Omogućava najveću konzistenciju i najmanju dostupnost od svih nivoa konzistencije. |
| EACH\_QUORUM | Vraća vrstu sa najnovijom vremenskom oznakom kada se dobije kvorum odgovora replika u svakom od centara podataka klastera. | Ista kao za LOCAL\_QUORUM. |
| LOCAL\_SERIAL | Isti kao za SERIAL, ali ograničen na centar podataka. | Ista kao za SERIAL. |
| LOCAL\_QUORUM | Vraća vrstu sa najnovijom vremenskom oznakom kada se kvorum kopija u centru podataka dobije od koordinatorskog čvora. Izbegava kašnjenje komunikacije između centara podataka. | Koristi se u više centara podataka uz NetworkTopologyStrategy i pravilno konfigurisani snitch. Ne uspeva kada se koristi SimpleStrategy. |
| LOCAL\_ONE | Vraća odgovor najbliže kopije, kao što je određeno snitch-em, samo ako je kopija u lokalnom centru podataka. | Ista kao u tabeli 1. |
| ONE | Vraća odgovor najbliže kopije, kao što je određeno snitch-em. Podrazumevano se u pozadini vrši ispravka pročitanih podataka, da bi i ostale kopije bile u konzistentnom stanju. | Omogućava najveću dostupnost u slučaju da se može tolerisati veća verovatnoća čitanja neažurnih podataka. |
| QUORUM | Vraća vrstu sa najnovijom vremenskom oznakom kada se dobije kvorum odgovora replika, ne uzimajući u obzir njihov centar podataka. | Omogućava visoku konzistenciju uz manju toleranciju na otkaze. |
| SERIAL | Dozvoljava čitanje trenutnog stanja podataka bez zahteva za dodavanjem ili ažuriranjem. Ako se pri SERIAL čitanju pronađe transakcija koja nije uneta, ona se unosi uz čitanje. | Lightweight transakcije. |
| TWO | Vraća najnovije podatke dve najbliže kopije. | Sličan nivou ONE. |
| THREE | Vraća najnovije podatke tri najbliže kopije. | Sličan nivou TWO. |

### O nivoima QUORUM i LOCAL\_QUORUM

Nivo QUORUM odgovara broju čvorova koji čine kvorum. Kvorum se računa i zaokružuje na ceo broj po sledećoj formuli:

gde je zbir\_faktora\_replikacije – zbir svih podešavanja pod nazivom replication\_factor za svaki centar podataka.

Na primer, za klaster sa pet centara podataka gde svaki ima faktor replikacije tri, vrednost QUORUM-a bi bila (5\*3/2) +1 = 8. U slučaju da je konzistencija prioritet, da bi se uvek pročitali ažurni podaci, treba da važi formula:

Za dati primer faktor replikacije bi morao biti manji od 8 + 8 = 16.

Da bi se izračunala vrednost LOCAL\_QUORUM nekog centra podataka, koristi se formula:

Podrazumevana vrednost za LOCAL\_QUORUM je (3/2) + 1 = 2 (zaokružena na ceo broj).

Koristeći LOCAL\_QUORUM, izbegava se kašnjenje zbog validacije podataka između više centara podataka, što bi bilo slučaj za QUORUM. Nivo konzistencije se ne menja prilikom dodavanja novih čvorova u klaster.

## Upis i čitanje u Cassandra čvorovima

Cassandra poseduje arhitekturu koja omogućava upis i čitanje iz bilo kog čvora, pa svaki od korisnika može da se poveže sa bilo kojim centrom podataka i da čita ili upiše podatke, gde pri upisu dolazi do automatskog particionisanja i repliciranja podataka u klasteru.

Dve najvažnije stavke kada je u piitanju određivanje kopija u svakom od centra podataka su da se čitanje može izvršiti lokalno, bez kašnjenja koje se javlja između centara podataka i scenariji otkaza. Najčešće se više centara podataka klastera definiše na sleće načine:

* *Dve replike u svakom od centara podataka* – omogućava toleranciju na otkaz jednog čvora po replikacionoj grupi, uz lokalna čitanja sa nivoom konzistencije *ONE*.
* *Tri replike u svakom od centara podataka* – omogućava toleranciju na otkaz jednog čvora po replikacionoj grupi, uz nivo konzistencije vrednosti *LOCAL\_QUORUM* ili više otkaza čvorova po centru podataka, koristeći nivo konzistencije *ONE.*

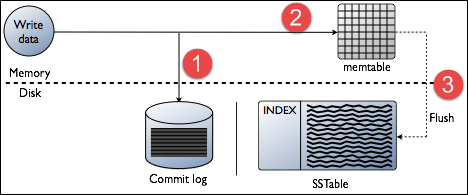
Takođe je moguće kreirati asimetrične replikacione grupe. Na primer, kada su u pitanju zahtevi aplikacije u realnom vremenu, koriste se tri kopije, dok se jedna kopija koristi na drugom mestu za analizu podataka. U nastavku će biti obrađene operacije upisa i čitanja.

### Operacija upisa

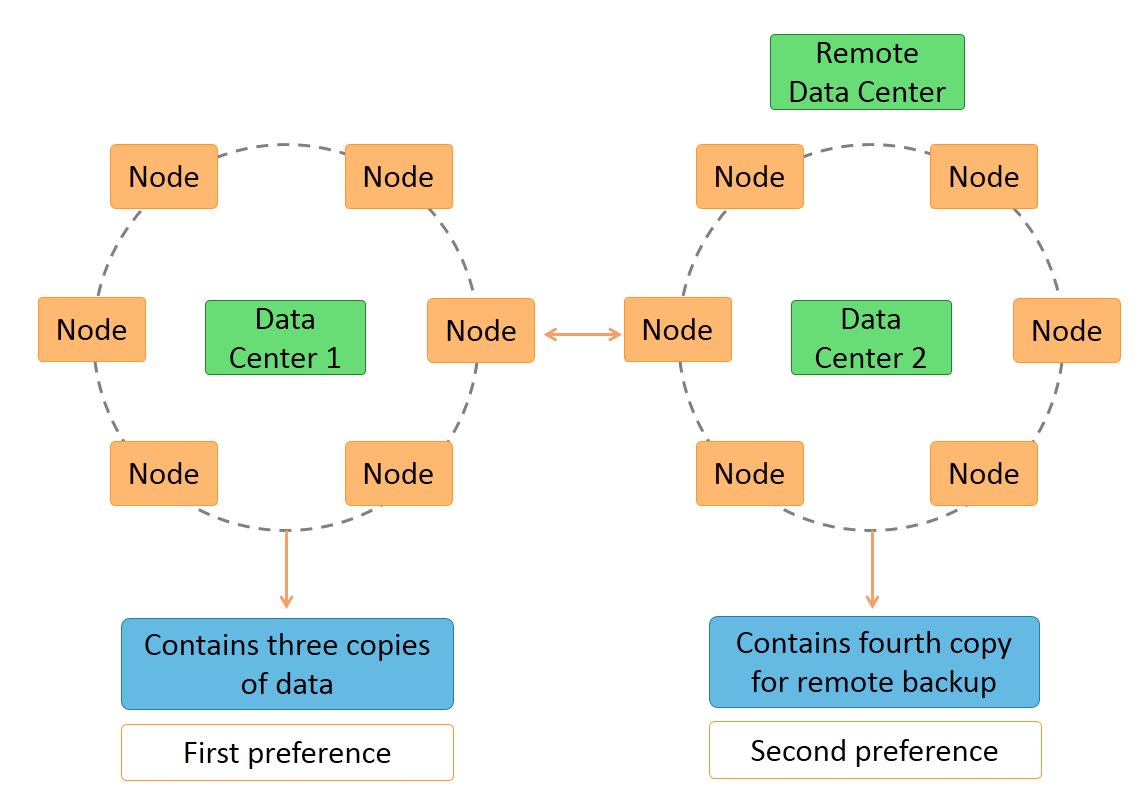
Upisi su brzi i sastoje se od sledećih koraka (Slika 15):

1. Koordinatorski čvor šalje zahtev za upisom replikama.
2. Ako su sve replike dostupne, primiće zahtev, bez obzira na svoj nivo konzistencije.
3. Pri dolasku zahteva za upisom u čvor, on se upisuje u commit log na disku. Commit log upisuje podatke transakcije i služi kao backup.
4. Slanje podataka čvoru pronađenom preko hash vrednosti.
5. Podaci sa čvorova se upisuju u memtable – tabelu čuvanu unutar memorije za trenutno čuvanje podataka.
6. Čvor šalje potvrdu uspeha upisa u slučaju uspešnog upisa u commit log i memtable.
7. Kada se memtable napuni, podaci se upisuju iz nje u sstable (Sorted String Table), u memoriji. Sstable sadrži podatke o svakom ažuriranju tabele.
8. Podaci u realnoj tabeli se ažuriraju preko sstable-a.

Na primer, u centru podataka sa faktorom replikacije tri, tri kopije primaju zahtev za upisom. U slučaju nivoa konzistencije koji iznosi jedan, samo će jedna od kopija poslati potvrdu upisa. Ako ostale dve kopije izgube podatke zbog problema u čvoru, ugrađeni mehanizam za oporavak će učiniti te kopije konzistentnim (Slika 16). Ako čvor odgovoran za podatke otkaže, oni se upisuju u privremeni čvor – tempnode. Privremeni čvor će privremeno prihvatiti podatke dok odgovorni čvor ne bude vraćen u funkciju.

[](https://www.guru99.com/images/cassandra/021116_0524_CassandraAr4.png)

Slika – upis podataka

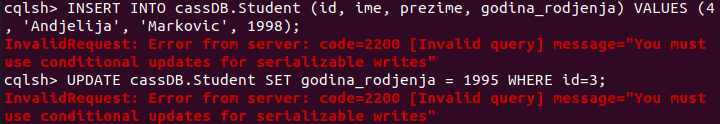


Slika – centri podataka sa svojim preferencijama

Pri podešavanju nivoa konzistencije upisa na SERIAL (Slika 17), može se uočiti da nije dozvoljen upis ili ažuriranje postojećih podataka (Slika 18).



Slika – podešavanje nivoa konzistencije



Slika – neuspešno ubacivanje i ažuriranje podataka

### Operacija čitanja

Čitanje u Cassandra-i se odvija paralelno u svim čvorovima, što ubrzava ovaj proces. U slučaju da čvor otkaže, podaci se čitaju iz svoje replike. Prioritet replike zavisi od njene udaljenosti:

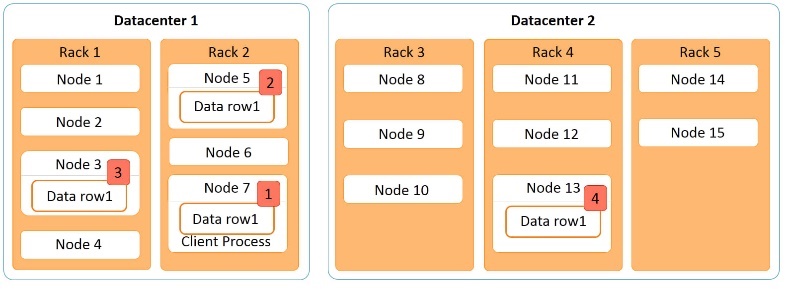
* *Lokalni podaci (data local)* imaju najveći prioritet i nalaze se na posmatranom čvoru.
* Podaci drugi po prioritetu se nalaze na istom rack-u u odnosu na posmatrani čvor i nazivaju se *rack local*.
* *Data center local* su podaci koji se nalaze na istom centru podataka.
* Najmanji prioritet imaju podaci iz drugog cenra podataka.

Podaci iz memtable-a i sstable-a se prvo proveravaju, da bi se čitanje ubrzalo, u slučaju da se već nalaze u memoriji. Postoje tri vrste zahteva za čitanjem koje koordinatorski čvor šalje kopijama:

1. Direktni zahtev
2. Digest zahtev
3. Read repair zahtev

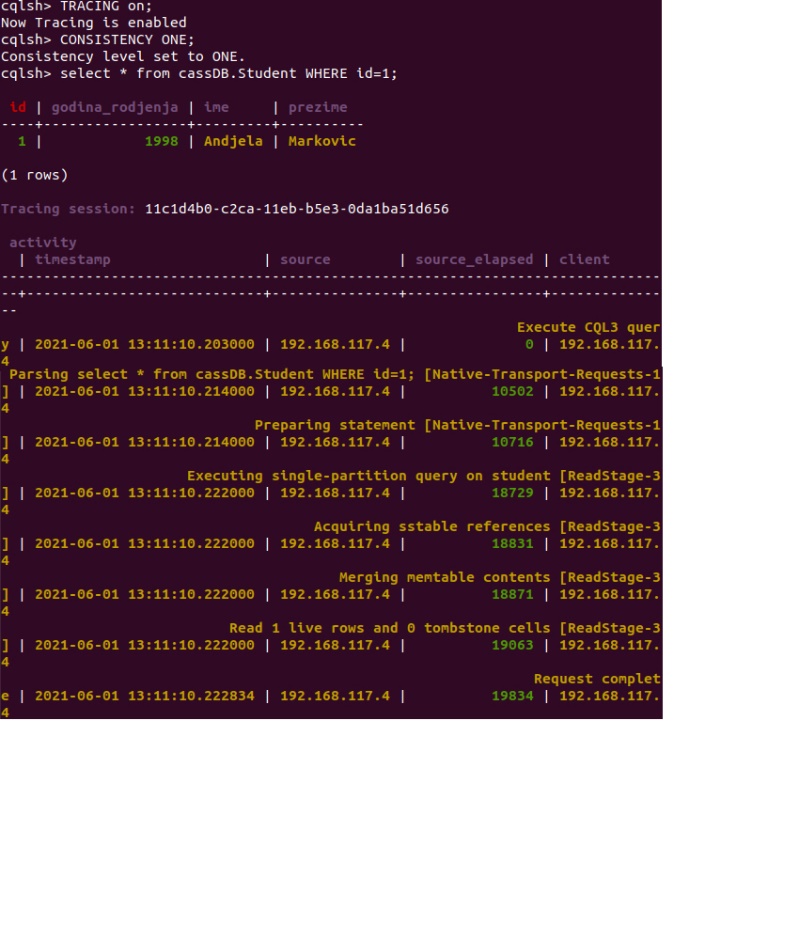
Koordinatorski čvor šalje direktni zahtev jednoj od kopija, a zatim šalje digest zahtev broju replika koji je određen nivoom konzistencije i proverava da li su vraćeni podaci ažurirani. Zatim koordinatorski čvor šalje digest zahtev preostalim replikama. Ako neki od čvorova vraća zastarelu vrednost, u pozadini se šalje zahtev za ispravkom pročitanih podataka (read repair).

U primeru sa slike 19 ilustrovan je proces čitanja. Original vrste smešten je u čvor 7, a njene kopije redom u čvorove 3, 5 i 13. Samo poslednja kopija je u drugom centru podataka. Ako klijentski proces pristupa sa čvora 7 originalu vrste, najveći prioritet ima taj čvor, jer su podaci lokalni. Sledeći po prioritetu je čvor 5, jer se nalazi u istom rack-u kao čvor 7. Zatim sledi čvor 3, pošto deli centar podataka sa datim čvorom, a najniži prioritet ima čvor 13, jer je u različitom centru podataka.



Slika 19 – centri podataka, čvorovi i rack-ovi sa kopijama podataka

U narednim primerima će biti praćeno čitanje jedne vrste tabele Student uz različite nivoe konzistencije. Najpre, kada je nivo konzistencije ONE, vraćaju se traženi podaci sa najbliže kopije, koja se u ovom slučaju nalazi na samom čvoru kome je pristupljeno (Slika 20).



Slika – praćenje izvršenja naredbe SELECT za nivo konzistencije čitanja ONE

Nivo konzisnencije TWO zahteva odgovor dva čvora (Slika 21), a nivo QUORUM takođe, jer QUORUM dva čvora iznosi 2 / 2 + 1 = 2 (Slika 22).

|  |  |
| --- | --- |
|  |  |
| Slika 21 – praćenje izvršenja naredbe SELECT za nivo konzistencije čitanja TWO | Slika 22 – praćenje izvršenja naredbe SELECT za nivo konzistencije čitanja QUORUM |

# Zaključak

Cassandra je baza podataka koja radi sa big data, dizajnirana tako da ne poseduje jednu tačku prekida. Ova baza ima prstenastu arhitekturu, pri čemu se uloge čvorova ne razlikuju. Takođe, podržava mrežnu topologiju sa više centara podataka, rack-ova i čvorova. Podaci se dele na particije zbog raspodele opterećenja u ovom distribuiranom sistemu. Particionisanje podataka je transparentno i vrši se na osnovu hash vrednosti ključeva.

Akcenat u ovom radu je bio na replikaciji podataka. Replikacija predstavlja kreiranje kopija podataka koje se smeštaju na različitim čvorovima, da bi se povećala otpornost na otkaze. Ona se može vršiti i u okviru više centara podataka. Osnovni faktori kod replikacije su faktor replikacije i strategija replikacije, koji određuju broj kopija i način njihove distribucije u sistemu. Ovo je jedna od najznačajnijih osobina Cassandra-e jer obezbeđuje da ne postoji jedna tačka prekida.

Čitanje i upis se izvršavaju veoma brzo. Pri tome se može odrediti nivo konzistencije, koji određuje koliko čvorova treba da ima validnu kopiju podatka, da bi upis ili čitanje bilo uspešno. Za međusobnu komunikaciju između čvorova koristi se gossip protokol.

Zbog jednostavnosti podešavanja parametara vezanih za replikaciju podataka i pouzdanosti koju ona pruža, Cassandra predstavlja dobro rešenje u primenama koje zahtevaju otpornost na otkaz i brzu konfiguraciju. Jedna od primena može biti čuvanje podataka o zdravlju korisnika uređaja za praćenje zdravlja.

# 

# Literatura

1. *What is Cassandra?*, dostupno na: <https://www.datastax.com/cassandra> (pristupljeno 25. 3. 2021)
2. *About Cassandra Replication Factor and Consistency Level*, dostupno na: Levelhttps://docs.apigee.com/private-cloud/v4.17.09/about-cassandra-replication-factor-and-consistency-level (pristupljeno 26. 4. 2021)
3. *Apache Cassandra Architecture Tutorial,* dostupno na: <https://www.simplilearn.com/cassandra-architecture-tutorial-video> (pristupljeno 30. 4. 2021)
4. *Data Replication*, dostupno na: <https://teddyma.gitbooks.io/learncassandra/content/replication/data_replication.html> (pristupljeno 30. 4. 2021)
5. *Data replication*, dostupno na: <https://docs.datastax.com/en/cassandra-oss/3.x/cassandra/architecture/archDataDistributeReplication.html> (pristupljeno 30. 4. 2021)
6. *Understanding Data Partitioning and Replication in Apache Cassandra*, dostupno na: <https://www.slideshare.net/DataStax/understanding-data-partitioning-and-replication-in-apache-cassandra> (pristupljeno 2. 5. 2021)
7. *How is the consistency level configured?*, dostupno na: <https://docs.datastax.com/en/dse/6.7/dse-arch/datastax_enterprise/dbInternals/dbIntConfigConsistency.html> (pristupljeno 1. 5. 2021)
8. *How to Setup a Cassandra Cluster*, dostupno na: <https://www.bmc.com/blogs/setup-cassandra-cluster/> (pristupljeno 1. 5. 2021)

1. Podaci koji svakodnevno nastaju na internetu u količinama koje se mere terabajtima, nemaju strogo definisanu strukturu i brzo se generišu. [↑](#footnote-ref-1)
2. Particioni ključ služi za raspodelu podataka između čvorova i može biti deo ili se poklapati sa primarnim ključem. [↑](#footnote-ref-2)
3. Command line shell, koji se koristi za interakciju sa Cassandra-om preko CQL-a (Cassandra Query Language). [↑](#footnote-ref-3)
4. Kada čvorovi sa kopijama ne mogu da prihvate promenu, čvorovi koordinatori koji pokušavaju da ažuriraju pomenute kopije čuvaju trenutne hint-ove na svom lokalnom fajl sistemu za kasniju primenu. [↑](#footnote-ref-4)
5. Nivo izolacije neke lightweight transakcije. [↑](#footnote-ref-5)
6. Cassandra-ine lightweight transakcije podrazumevaju da nije dozvoljen eksplicitan commit ili rollback i nije garantovana atmičnost u okviru particija. Omogućeno je samo jednostavno proveravanje ograničenja transakcija. [↑](#footnote-ref-6)