

Univerzitet u Nišu

Elektronski fakultet

Seminarski rad

**Klasterovanje kod Neo4j baze podataka**

Sistemi za upravljanje bazama podataka, 2020/2021.

Mentor:

doc. dr Aleksandar Stanimirović

Student:

Vladimir Janjić, 1283

**Sadržaj**

[Uvod 3](#_Toc75709318)

[Arhitektura i mehanizmi rada klastera 4](#_Toc75709319)

[Serveri jezgra 4](#_Toc75709320)

[Replike za čitanje 5](#_Toc75709321)

[Održavanje konzistencije 6](#_Toc75709322)

[Mehanizmi rada 7](#_Toc75709323)

[Rutiranje zahteva za upis i čitanje 9](#_Toc75709324)

[Keširanje podataka 11](#_Toc75709325)

[Upravljanje klasterom 12](#_Toc75709326)

[Formiranje klastera 12](#_Toc75709327)

[Rad sa podacima 18](#_Toc75709328)

[Zaključak 21](#_Toc75709329)

[Literatura 22](#_Toc75709330)

# Uvod

Potrebe većine aplikacija, koje opslužuju više korisnika istovremeno, obično prevazilaze mogućnosti jednog računara. Problem se javlja u ograničenim resursima koje jedan računar poseduje, kao i u mogućnošću da opsluži više zahteva, koji neretko dolaze sa veoma udaljenih lokacija. Za rešenje problema, postoje dva različita pristupa: jedinstveni računar sa ogromnim resursima (vertikalno skaliranje) ili više računara koji rade zajedno (horizontalno skaliranje).

Prvi pristup je realizovan upotrebom superračunara. Superračunari su projektovani tako da prevazilaze mogućnosti personalnih računara i uglavnom se koriste za aplikacije koje služe za intenzivna izračunavanja. Zbog svoje ogromne cene i prostora koji zauzimaju, koriste ih isključivo veoma bogate kompanije, za izračunavanja u raznim oblastima fizike i meteorologije.

Drugi pristup se sastoji u upošljavanju većeg broja slabijih računara koji rade na zajedničkom zadatku i poznat je po nazivom klasterovanje računara. Najčešće se radi o istovetnim računarima koji pokreće isti operativni sistem, ali je napretkom tehnologije omogućena upotreba različitih kombinacija hardvera i softvera. Računari takođe ne moraju da budu fizički na istoj lokaciji, kao što je slučaj kod superračunara, ali su i dalje vidljivi korisniku kao jedinstveni sistem. Upravo zbog toga se za ovakav pristup opredeljuje najveći broj kompanija čije aplikacije zahtevaju rad sa korisnicima na većem geografskom područiju.

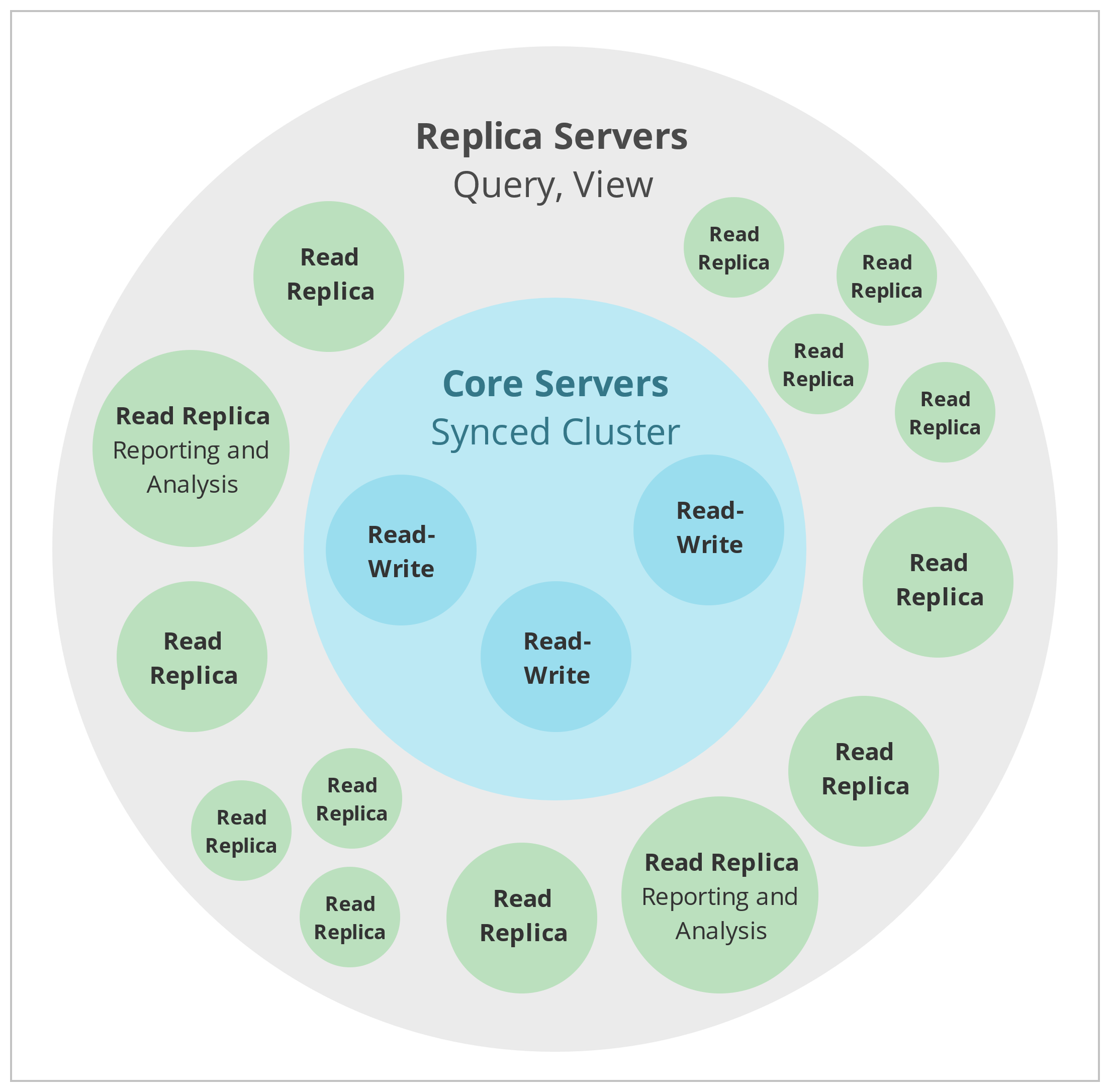
Postojanjem klastera postiže se veliki broj pogodnosti: velika propusnost aplikacije, redundantnost podataka, velika dostupnost... Povećana je i otpornost na greške, a takođe i omogućeno dodatno proširenje (ukoliko se za to ukaže potreba). Jedna od baza podataka koja podržava ovakvu upotreba jeste graf baza podataka Neo4j.

Klasterovanje je kod Neo4j baze podataka definisano kao „kauzalno klasterovanje“ (eng. *Causal Clustering*). Reč „kauzalno“ se odnosi na način na koji je aplikaciji omogućena konzistencija – pri čitanju, klijentskoj aplikaciji je zagarantovano čitanje bar onih upisa koje je sama izvršila.

U nastavku rada biće prikazani koncepti i način rada klasterovanja kod Neo4j baze podataka. Teorijski deo će obraditi arhitekturu, mehanizme i protokole koji omogućavaju funkcionalnost. U praktičnom delu rada će biti reči o samoj upotrebi klasterovanja – od kreiranja klastera, preko dodavanja odn. brisanja članova, do podešavanja konfiguracije i izbegavanja potencijalnih problema.

# Arhitektura i mehanizmi rada klastera

Klasterovanje kod Neo4j baze podataka zasniva se na konceptu „master-slave“, koji je modifikovan za potrebe konkretne baze. Na osnovu uloga i funkcija koje obavljaju, izdvajaju se dva tipa servera (ili čvorova): serveri jezgra (eng. *Core Servers*) i replike za čitanje (eng. *Read Replicas*). Na slici 1 prikazan je izgled arhitekture kod upotrebe klasterovanja.



Slika 1. Arhitektura kauzalnog klasterovanja.

## Serveri jezgra

Serveri jezgra, kao što im ime sugeriše, predstavljaju samu srž sistema. Glavna uloga im je u čuvanju podataka i u osiguravanju ispravnosti istih. Učestvuju u čitanju i pisanju podataka i svaki sadrži potpunu kopiju podataka. Broj servera zavisi od potrebne odpornosti na greške. Prema Raft protokolu, broj servera koji mora da potvrdi upis jeste n/2 + 1, gde je broj n broj servera jezgra. Drugim rečima, potrebna je potvrda većinskog broja servera pre nego što baza podataka vrati odgovor klijentu koji je zahtevao upis. Prateći ovu formulu, najmanji broj servera koji je potreban da bi otpornost na greške bila omogućena jeste tri. Ovaj broj servera dozvoljava otkazivanje samo jednog servera uz nastavak rada, dok otkazivanje dva postavlja sistem u mod za čitanje. Prema tome, za povećanje otpornosti na još jedan server, potrebno je dodati dva nova (otpornost na dva za pet servera, tri za sedam...). Iako povećanje broja servera utiče na povećanje otpornosti na greške, takođe se javlja i povećano kašnjenje u odgovoru jer veći broj servera mora potvrditi upis.

Svi serveri jezgra vidljivi su klijentu kao jedinstvena celina. Međutim, interno postoji razlika među serverima u odnosu na ulogu prihvatanja upisa. Po specifikaciji Raft protokola, postoji zapis obavljenih upisa, tako da se nove izmene ne kopiraju direktno, već se kopira zapis koji omogućava izvršenje istovetnih operacija na drugom serveru koji je poslao zahtev. Za potrebe ovakvog ponašanja izdvojene su uloge, tj. postoje vođa (eng. *Leader*) i pratioci (eng. *Followers*), odn. glavni server i sporedni serveri. Glavni server ima ulogu u primanju, rutiranju i inicijalnom obavljanju upisa. Nakon što je upis obavljen na glavnom serveru, ostali serveri asinhrono zahtevaju kopiju zapisa kako bi mogli da obave istovetnu operaciju. Zadovoljavanjem zahteva za većinom, vraća se odgovor klijentu i baza podataka je spremna za sledeći upis.

Kod ovakve postavke servera, može se javiti problem pada glavnog servera. Ukoliko nakon pada i dalje postoji većina servera, neki od preostalih može pokrenuti proces izbora. Server koji je pokrenuo izbore preuzima ulogu kandidata (eng. *Candidate*) i ostali serveri glasaju za ili protiv da kandidat postane novi vođa. Najčešće za vođu bude odabran server sa najsvežijim zapisom transakcija. Ovaj proces se drugačije naziva i automatskim premošćavanjem grešaka. Korisnik nema informaciju o izborima kao ni o trenutnom vođi. Ukoliko otkaže veći broj servera i ne postoji većina, baza podataka prelazi u mod eksluzivnog čitanja (eng. *Read-only*).

Određivanje uloga vrši se na nivou baze podataka, a ne instanci servera. To znači da jedan isti uređaj može da u jednoj bazi podataka bude vođa, a u drugoj pratilac. Ovakva implementacija izvršena je u cilju raspoređivanja opterećenja, tako da svaki od servera jezgra učestvuje približno jednako u upisima. Još jedan mehanizam koji se koristi u ove svrhe jeste automatsko prebacivanje uloge vođe. Pri početnom kreiranju klastera, uvek postoje barem dve baze podataka: „system“ i „neo4j“.

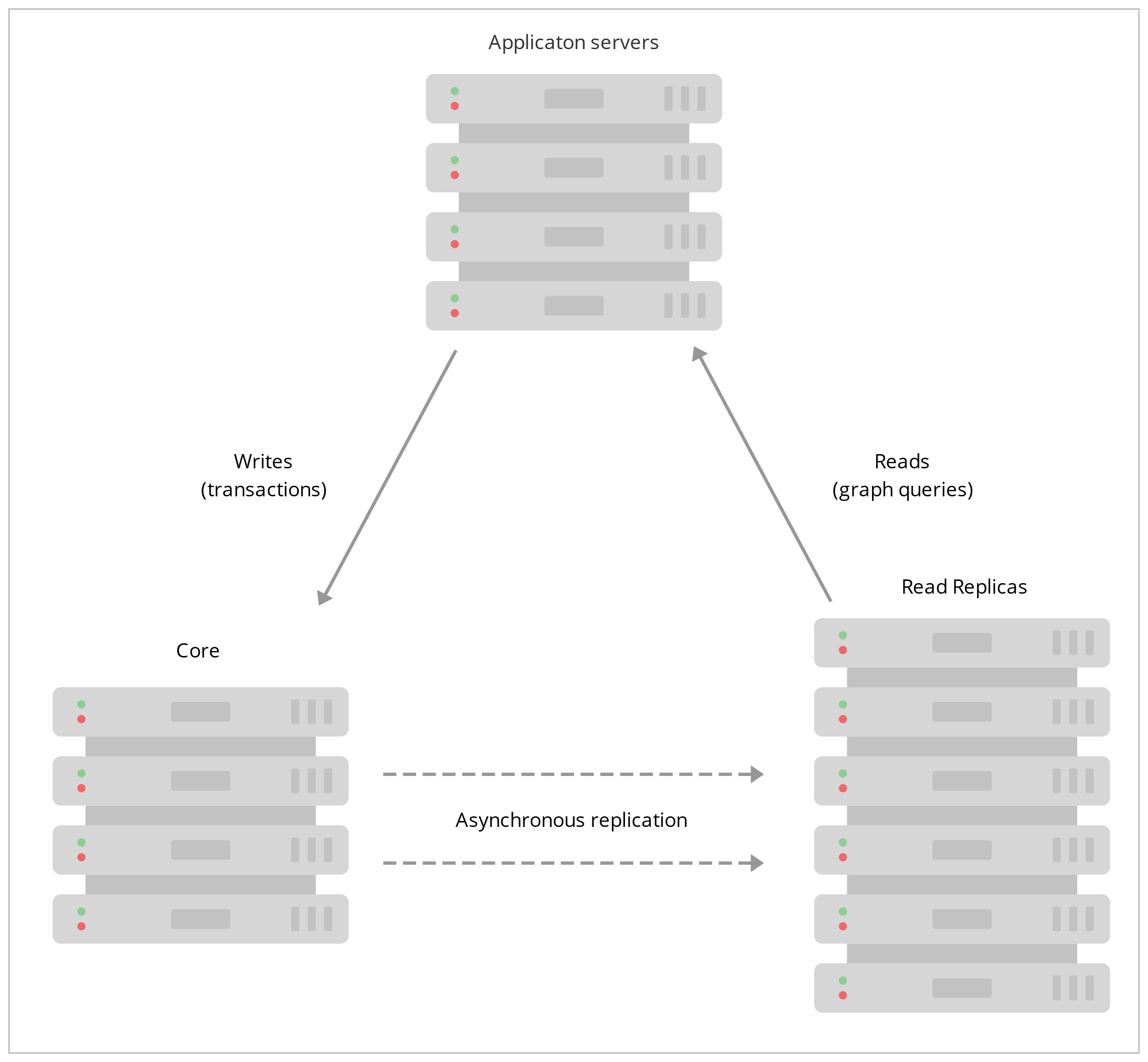
Korišćenjem Bolt protokola, omogućeno je prihvatanje upita od strane servera koji ga neće izvršiti, tj. servera koji nije vođa za potrebnu bazu podataka. Ukoliko je ispravno obavljeno podešavanje, izvršiće se rutiranje na strani servera do servera koji može da izvrši zahtev.

## Replike za čitanje

Drugi tip servera koji se koristi kod Neo4j arhitekture klauzalnog klasterovanja jesu replike za čitanje. Glavna uloga ovih servera jeste u skaliranju čitanja. Za razliku od servera jezgra, njihov broj može biti veoma veliki, zbog toga što pad replika ne utiče na stabilnost rada jezgra. Održavanje konzistencije podataka odvija se preko zapisa transakcija. Replike zahtevaju novonastale izmene od nekog servera jezgra, nakon čega ažuriraju svoje stanje. Veliki broj replika može biti opsluženo jednim serverom jezgra. Funkcije replike i jezgra ne bi trebalo menjati (prebacivanje iz jezgra u repliku i obrnuto).

## Održavanje konzistencije

Kao što je prethodno napomenuto, arhitektura klasterovanja kod Neo4j baze podataka zasniva se na principu kauzalne konzistencije. To znači da pri postojanju više servera, svaki održava ispravan redosled obavljenih transakcija, tako da čitanje sa bilo kog servera daje iste rezultate, bilo da se radi o serveru jezgra ili nekoj od replika. Za održavanje redosleda, klijent može zahtevati marker koji nalaže klasteru da izvrši transakciju samo na onim serverima na kojima je izvršena prethodna transakcija klijenta. Na slici 2, prikazan je upis nad serverom jezgra, nakon čega se ista izmena može pročitati i sa replike.

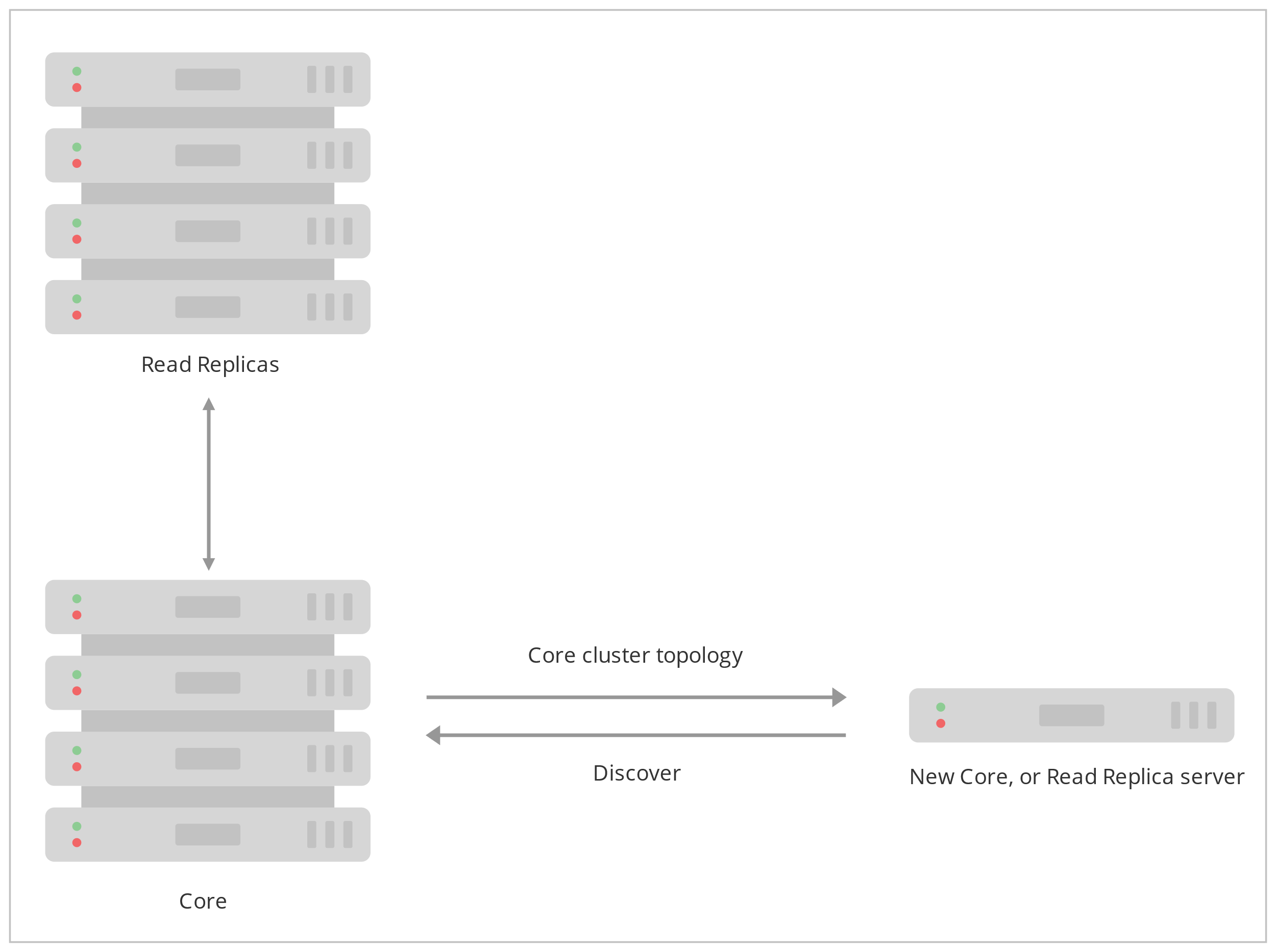


Slika 2. Održavanje kauzalne konzistencije.

Kada je konzistencija u pitanju, treba pomenuti i ponašanje klastera u odnosu na ACID osobine i CAP teoremu. Kod jedne instance Neo4j servera, ACID je zagarantovan. Međutim, zahtev za konzistencijom (i eventualno dostupnošću) se mora ublažiti zbog dodatnog kašnjenja koje unosi sinhronizacija. Prema tome, može se desiti da pri čitanju, neke od poslednjih izmena ne budu odmah vidljive. Ovakvo ponašanje svrstava Neo4j bazu podataka kao AP bazu (Availability/Partition tolerance).

## Mehanizmi rada

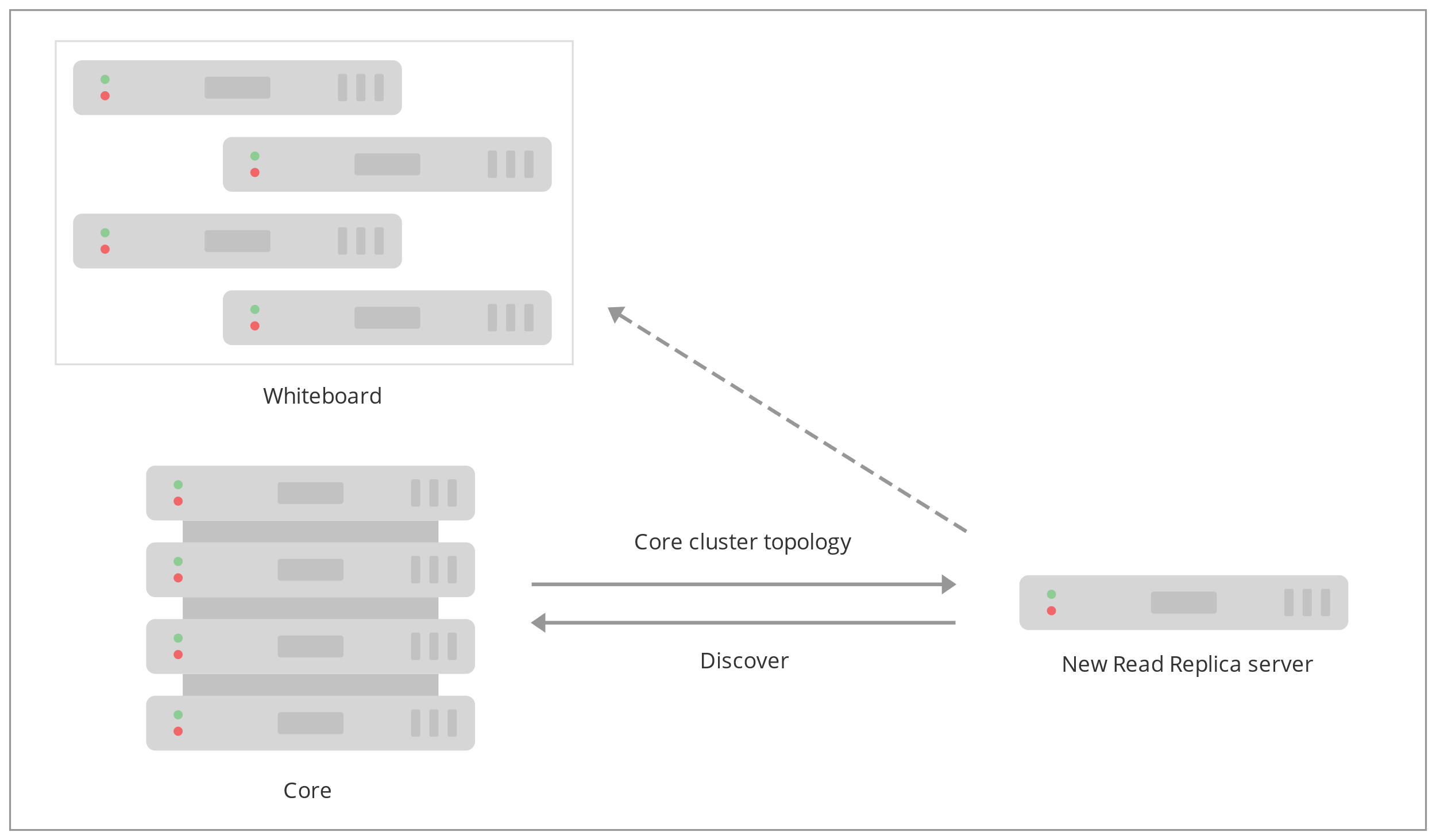
Pri kreiranju novog klastera, aktiviraju se protokoli koji su potrebni za normalan rad servera članova. U okviru konfiguracije svakog od servera, sadržana je lista članova koji čine početno jezgro. Nakon pokretanja servera, protokol za otkrivanje (eng. *Discovery protocol*) ima ulogu u povezivanju svih članova navedenih u sekciji za početno stanje klastera. Prvi server koji postane dostupan vrši kreiranje samog klastera, dok se ostali priključuju dobijanjem informacija o topologiji klastera. Svaki novi server, koji nije obuhvaćen inicijalnim podešavanjem, takođe mora pokrenuti protokol za pristup klasteru, bilo da se radi o serveru jezgra ili replici. Na slici 3 prikazan je princip komunikacije servera.



Slika 3. Komunikacija servera pri otkrivanju novih servera.

Pristup novog člana jezgru zahteva registrovanje i izmene u Raft zapisu. Zapis se prosleđuje ostalim članovima na potvrdu i tek kada većina primi zahtev, novi server postaje zvanično član klastera. Novi server može pristupiti kao prazan ili može sadržati istovetne baze podataka kao i ostali članovi (samo sa ranijim verzijama). Ako se kreće sa sinhronizacijom ispočetka, novi server može ostati nedostupan duži vremenski period dok se ne dovede u konzistentno stanje. Ukoliko je postojala ranija verzija neke od baza podataka, izvršiće se samo novonastale izmene iz zapisa transakcija.

Dodavanje replika ne zahteva rad Raft protokola, već se replike dodaju u deljenu „tablu“ (eng. *Whiteboard*). Kako ne utiču na topologiju klastera, njihov broj može biti višestruko veći od broja servera jezgra. Stanje „table“ se ažurira pri svakom dodavanju, odn. uklanjanju replika. Uloga „table“ jeste u rutiranju zahteva za čitanje. Na slici 4 prikazano je dodavanje nove replike u okviru deljene „table“.

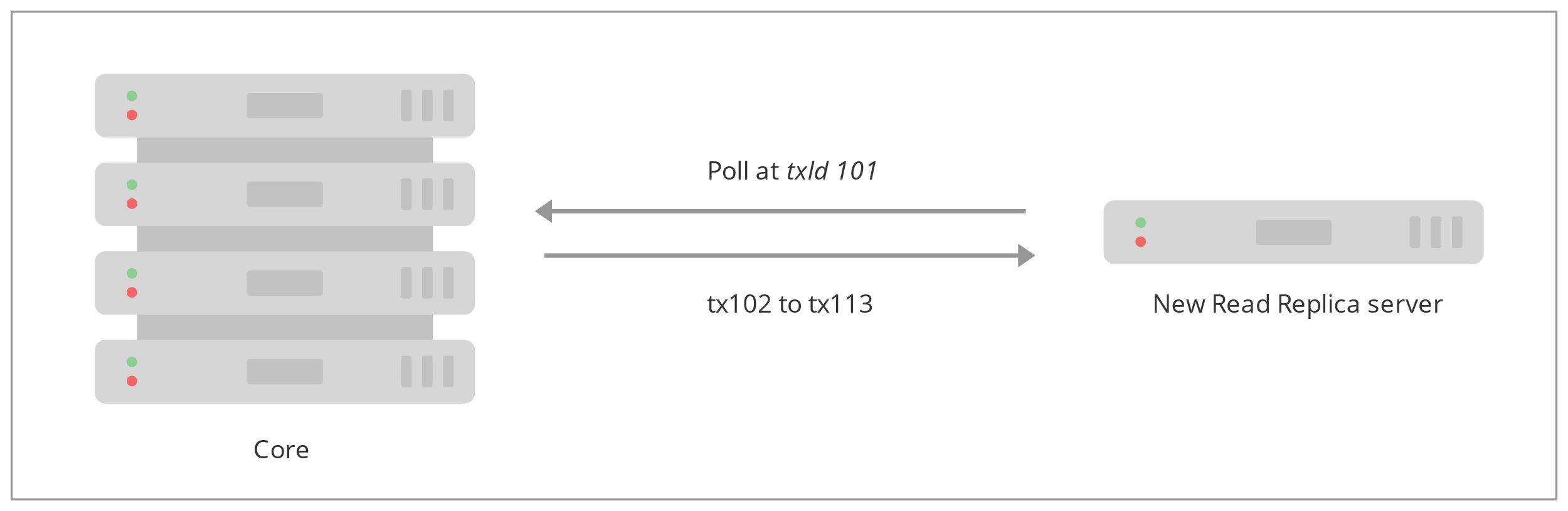


Slika 4. Dodavanje novih replika za čitanje.

Normalan rad klastera odvija se putem Raft protokola i transakcionog zapisa. Svaki od vođa baza podataka (samo jedan po bazi podataka) učestvuje u primanju i izvršavanju upisa. Zatim se vrši upis na vrh transakcionog zapisa i prosleđivanje stavke ostalim serverima. Vođa ima informaciju o tome na koliko je servera uspešno izvršen upis. Kada se postigne upis na većini servera, vrši se potvrda stavke u zapisu i izdaje odgovor klijentu koji je zahtevao upis.

U okviru normalnog rada klastera, pad nekog od servera smatra se normalnom pojavom. Pad pratilaca ne nalaže nikakve izmene u radu (dok god postoji većina), dok pad vođe zahteva izbor novog. Novi vođa se bira po kriterijumima najdužeg mandata, najdužeg zapisa i najsvežije potvrđene stavke zapisa. U osnovnom podešavanju, izbor se vrši nakon 20 sekundi od primećenog pada vođe. Pri tom treba napomenuti da ne dolazi do gubitka podataka jer ostali serveri i dalje čuvaju istovetne kopije.

S druge strane, replike za čitanje se smatraju nepostojanim resursom. To znači da je predviđeno dodavanje i uklanjanje replika u zavisnosti od potreba, bez prevelikog uticaja na sam klaster. Takvo ponašanje utiče na to da replike imaju podatke koji su u manjoj ili većoj meri zastareli. Priključivanjem replike, vrši se proces ažuriranja (eng. *Catchup protocol*) preko transakcionog zapisa. Rad procesa može se videti na slici 5.



Sliak 5. Prikaz rada „catchup“ protokola.

Problem može nastati kada nova replika previše zaostane za trenutnim stanjem klastera. U tom slučaju, nije moguće pronaći jedinstveni trenutak sinhronizacije od kog će se nastaviti ažuriranje. Jedino što preostaje replici jeste da vrši direktno kopiranje podataka sa nekog od servera jezgra. Time se vrši dodatno opterećenje servera jezgra proporcionalno količini podataka koje je potrebno kopirati. Ekstremni slučaj problema nastaje kada kopiranje podataka traje previše dugo, tako da replika i nakon kopiranja previše zaostaje, što inicira novu kopiju i tako u krug, sve do intervencije operatera. S tim na umu, treba obratiti pažnju na svežinu podataka novih replika.

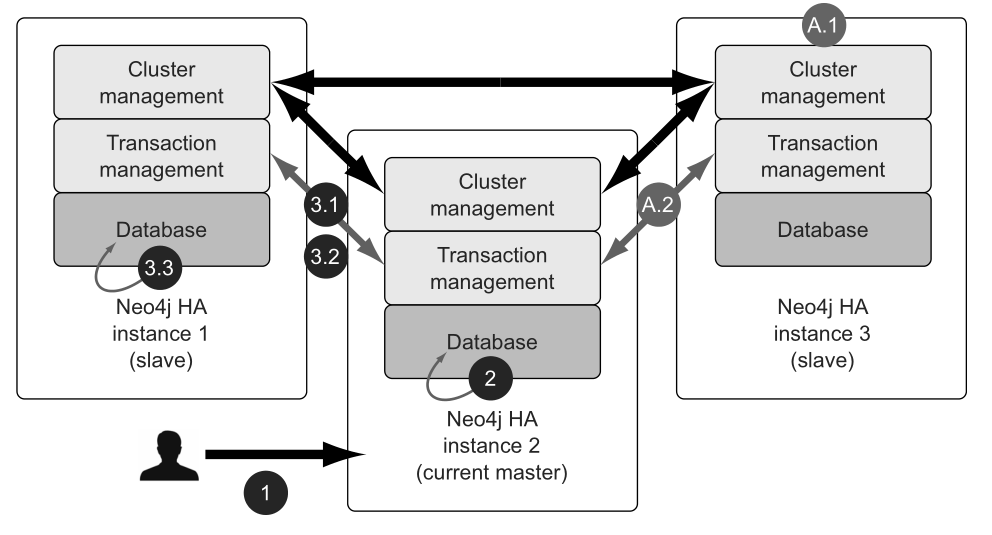
Gašenjem replike, oslobađa se prostor u memoriji table koje je zauzimala. Replika se uklanja iz „table“ i ukoliko ostane nedostupna na neko vreme. Ako je pri tom doslo do gubitka podataka ili transakcionog zapisa, vrši se „premotavanje unazad“ (eng. *Rollback*) do konzistentnog stanja.

U gašenju nekog od servera jezgra, potrebno je uključivanje Raft protokola koji vrši uklanjanje servera iz grupe. Ako je uklonjeni server bio vođa, vrši se izbor novog. Klaster nastavlja normalno sa radom dok god postoji većina. Pogodnost kod Raft protokola ogleda se u automatskom smanjivanju veličine klastera. Ovakvo ponašanje je omogućeno podešavanjem konfiguracije, i moguće je samo uz postepeno isključivanje servera da bi Raft mogao da prilagodi klaster novom okruženju. Smanjivanjem veličine klastera produžava se rad sistema, ali se pri tom smanjuje otpornost na greške (eng. *Fault tolerance*).

## Rutiranje zahteva za upis i čitanje

U okviru „master-slave“ arhitekture, pisanje je moguće obaviti samo na glavnom serveru, odn. master-u. Međutim, najčešće ne postoji ograničenje koji će od servera primiti zahtev, bilo da se radi o pisanju ili čitanju. Kod Neo4j baze podataka, mogu se izdvojiti dva različita procesa izvršenja upita, u zavisnosti od toga da li zahtev prima server koji će izvršiti upit, ili ne.

Prvi proces obuhvata primanje i izvršenje upita na jednom serveru, i to na master-u, odn. vođi Raft grupe. Na slici 6 prikazani su koraci u izvršenju upita kad zahtev prima vođa.

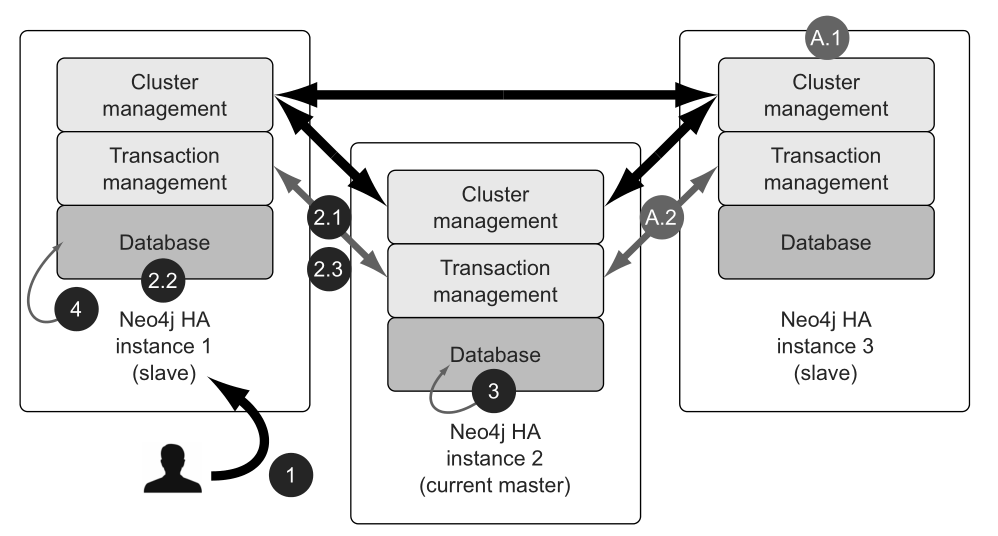


Slika 6. Koraci u izvršenju upita koji prima vođa grupe.

Koraci u izvršenju su sledeći:

1. Klijent izdaje zahtev.
2. Vođa zaključava bazu i vrši transakciju kao da se radi o samostalnom serveru.
   1. Vođa prosleđuje izmene pratiocima (broj određen konfiguracijom). Ukoliko ne uspe u tome, transakcija je i dalje validna.
   2. Pratioci ažuriraju sebe ukoliko nemaju sveže podatke pre novonastale izmene.
   3. Pratioci izvršavaju transakciju.

Drugi proces obuhvata primanje zahteva na nekom od pratioca koji potom prosleđuju isti vođi radi izvršenja. Na slici 7 prikazani su koraci u izvršenju upita kada zahtev prima pratilac.



Slika 7. Koraci u izvršenju upita koji prima neki od pratioca.

Koraci u izvršenju su sledeći:

1. Klijent izdaje zahtev.
   1. Zaključavanje baze podataka vođe.
   2. Zaključavanje baze podataka pratilaca.
   3. Vrši se ažuriranje pratilaca, ukoliko podaci nisu sveži.
2. Zahtev se izvršava na vođi.
3. Ukoliko je zahtev uspešno izvršen na vođi, izvršava se i na pratiocu.

Postojanje mogućnosti za prosleđivanje zahteva svim serverima ima svoje prednosti, ali i mane:

* Pisanje preko pratilaca usporava sistem zbog potrebne sinhronizacije servera pre izvršenja upita.
* Pisanjem preko pratilaca poboljšava se izdržljivost baze podataka zbog toga što se osigurava postojanje bar dveju kopija podataka sa novonastalim izmenama. Drugo rešenje jeste pisanje na vođi uz prosleđivanje upisa na određen broj pratioca pre potvrđivanja zahteva.
* Potrebno korišćenje eksternih hardverskih ili softverskih balansera opterećenja ukoliko je omogućeno forsiranje određenih upisa na odgovarajuće servere (upisi generalno preko vođe, čitanje generalno preko replika za čitanje).

Zahtevi za čitanje su jednostavniji od zahteva za upisom i izvršavaju se na serveru koji je primio zahtev, bilo da se radi o serveru jezgra ili nekoj od replika za čitanje.

## Keširanje podataka

Do verzije 4.0, Neo4j baza podataka nije podržavala particionisanje podataka. Problem je bio u samoj graf prirodi baze podataka. Zbog toga se umesto particionisanja koristila tehnika „particionisanja keša“. Razlika se sastoji u tome što je deljenje keša zapravo tehnika za rutiranje. Korišćenjem nekog balansera opterećenja, zahtevi za istim podacima šalju se prema istim instancama. Na taj način je omogućeno da isti podaci ostanu duže vreme u RAM, čime se ubrzava odgovor na zahtev.

Druge strategije za rutiranje sastoje su se u poznavanju karakteristika podataka ili prosleđivanju zahteva na osnovu lokacije klijenta.

Od verzije 4.0, rešenje za particionisanje podataka ogleda se u postojanju Fabric-a.

# Upravljanje klasterom

Zaključno sa verzijom 3.5, upotreba klasterovanja bila je moguća samo uz kupovinu licence na nivou preduzeća (eng. *Enterprise Edition*). Međutim, zbog popularnosti tehnologije, došlo je do izmena u načinu licenciranja, tako da je u novijim verzijama (trenutna verzija 4.3.1) moguće isprobati klasterovanje potpuno besplatno na 30 dana, ili duže, ukoliko se koristi u svrhe obuke. Najnovija verzija (21. jun 2021.) radi isključivo sa Java JDK 11.

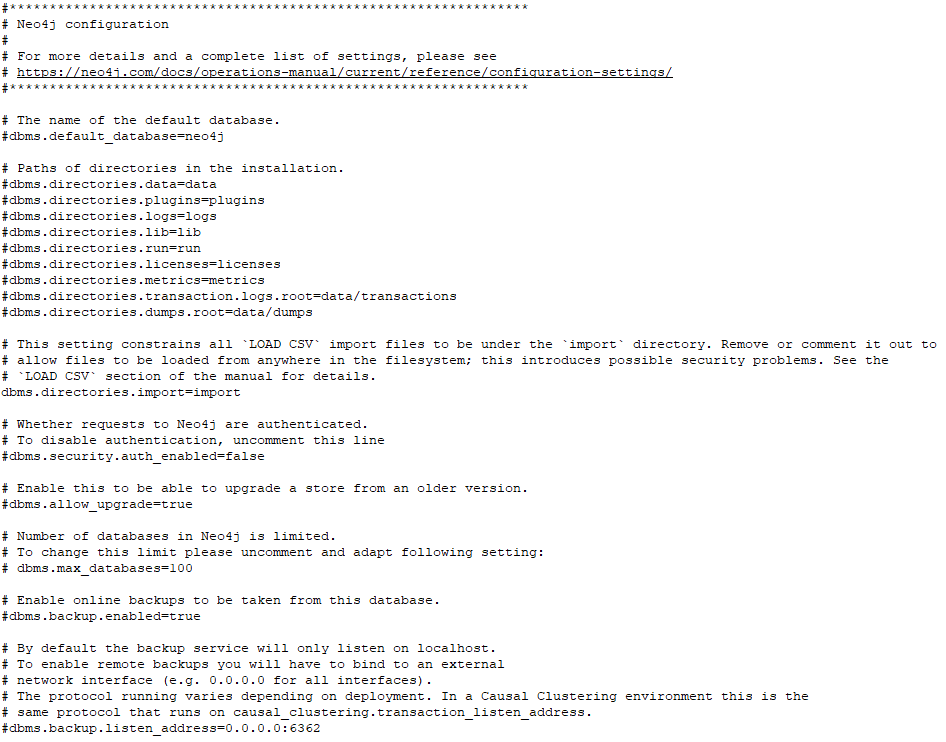
## Formiranje klastera

Kreiranje najjednostavnijeg klastera sastoji se u pokretanju tri instance servera jezgra. Međutim, dodavanjem još tri replike za čitanje, dodatno se poboljšava propusna moć i znatno smanjuje opterećenje jezgra, jer se čitanje može obavljati isključivo na replikama, dok serveri jezgra ostaju slobodni za operacije upisa.

Pre pokretanja servera, potrebno je izvršiti konfiguraciju samog klastera, kao i pojedinačnih servera. Celokupna konfiguracija nalazi se u poddirektorijumu **conf/neo4j.conf** svakog servera. Sadrži sledeća podešavanja:

* **Osnovna** (ime baze, putanje, autentifikacija, maksimalni broj baza, ...)
* **Memorijska** (Java memorija, keš, memorija za tansakcije – pojedinačno i ukupno, ...)
* **Mrežna** (IP adrese, Bolt protokol, rutiranje, ...)
* **SSL** (polise, putanje do polisa, Bolt/Https/Cluster/Backup SSL konfiguracija, ...)
* **Zapisa** (da li je omogućeno, veličina zapisa i stavki, tip zapisa, memorija, ...)
* **Kauzalnog klastera** (tip servera, adresa i port, podešavanja Raft protokola, ...)
* **Balansera opterećenja** (podešavanje dodataka za balansiranje opterećenja)
* **Dodatna podešavanja kauzalnog klastera**
* **Sigurnosna** (autentifikacija, autorizacija, TTL keš, ...)
* **LDAP** (podešavanje provajdera servisa i način komunikacije)
* **LDAP autentifikacija**
* **LDAP autorizacija**
* **Razna** (eng. *Miscellaneous*)
* **JVM** (parametri za Java virtuelnu mašinu i dodatna podešavanja)
* **Servisa** (ukoliko se Neo4j baza podataka korisit kao servis – DBaaS)
* **Druga...**

Na početku, većina podešavanja je zakomentarisana i ostavljeno je korisniku da vrši izmene tamo gde je potrebno. Svako od podešavanja takođe prati i dodatno objašnjenje. Na slici 8 dat je prikaz dela konfiguracionog fajla sa osnovnim podacima.



Slika 8. Izgled dela konfiguracionog fajla sa osnovnim podešavanjima.

Podešavanje klastera kreće osposobljavanjem servera koji će biti njegov prvi član. Za konkretanu upotrebu (tri servera jezgra i tri replike lokalno), potrebno je izmeniti sledeća podešavanja prikazana u tabeli 1:

|  |  |
| --- | --- |
| Podešavanje | Opis podešavanja |
| dbms.mode=CORE | Postavljanje servera u režim jezgra. |
| causal\_clustering.minimum\_core\_cluster\_size\_at\_formation=3 | Najmanji broj servera jezgra pri kreiranju klastera. Klaster odpočinje sa radom tek kada se priključi dovoljan broj servera. |
| causal\_clustering.minimum\_core\_cluster\_size\_at\_runtime=3 | Najmanji broj servera jezgra u toku rada. Kod većih servera, moguće je dozvoliti smanjivanje klastera za produžetak rada. Ukoliko broj dostupnih servera padne ispod navedenog broja, klaster prelazi u mod za ekskluzivno čitanje. |
| causal\_clustering.initial\_discovery\_members=localhost:5000,localhost:5001,localhost:5002 | Adrese članova koji inicijalno čine klaster. |
| causal\_clustering.discovery\_listen\_address=:5000 | Port za razmenu informacija o topologiji klastera, „discovery“ protokol za otkrivanje. Podrazumevano 0.0.0.0. |
| causal\_clustering.transaction\_listen\_address=:6000 | Port za osluškivanje nadolazećih transakcija. Podrazumevano 0.0.0.0. |
| causal\_clustering.raft\_listen\_address=:7000 | Port koji koristi Raft protokol (npr. za izbore i glasanje, razmena informacija o ulogama u Raft grupi). Podrazumevano 0.0.0.0. |
| dbms.connector.bolt.listen\_address=:7687 | Port koji koristi Bolt protokol (jedna od uloga je rutiranje zahteva na strani servera). Podrazumevano 0.0.0.0. |
| dbms.connector.http.listen\_address=:7474 | Port za http. Podrazumevano 0.0.0.0. |
| dbms.connector.https.listen\_address=:6474 | Port za https. Podrazumevano 0.0.0.0. |
| dbms.backup.listen\_address=0.0.0.0:6362 | Adresa za online backup. |

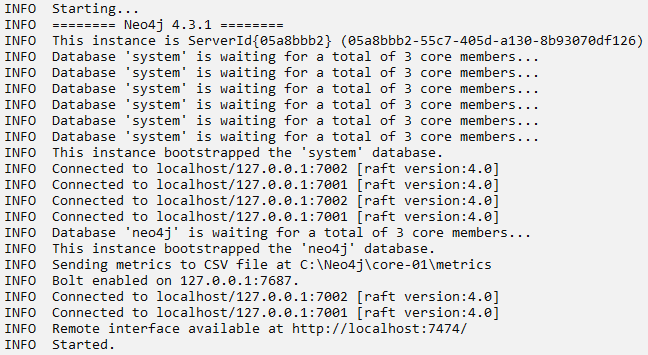
Tabela 1. Osnovna podešavanja prvog člana klastera.

Pokretanjem servera u režimu konzole, ispisuje se poruka o čekanju na minimalan broj članova pre nego što klaster bude u upotrebi. Zbog toga je potrebno obaviti konfiguraciju nad preostala dva servera jezgra, a zatim ih pokrenuti. Konfiguracioni fajlovi se neznatno razlikuju i prikaz korišćenih podešavanja dat je u tabeli 2. Treba napomenuti da je pre navedenih izmena izvršeno kopiranje konfiguracije sa prvog servera, kako bi se smanjio obim posla.

|  |  |
| --- | --- |
| Podešavanja servera jezgra 2 | Podešavanja servera jezgra 3 |
| causal\_clustering.discovery\_listen\_address=:5001 | causal\_clustering.discovery\_listen\_address=:5002 |
| causal\_clustering.transaction\_listen\_address=:6001 | causal\_clustering.transaction\_listen\_address=:6002 |
| causal\_clustering.raft\_listen\_address=:7001 | causal\_clustering.raft\_listen\_address=:7002 |
| dbms.connector.bolt.listen\_address=:7688 | dbms.connector.bolt.listen\_address=:7689 |
| dbms.connector.http.listen\_address=:7475 | dbms.connector.http.listen\_address=:7476 |
| dbms.connector.https.listen\_address=:6475 | dbms.connector.https.listen\_address=:6476 |
| dbms.backup.listen\_address=0.0.0.0:6363 | dbms.backup.listen\_address=0.0.0.0:6364 |

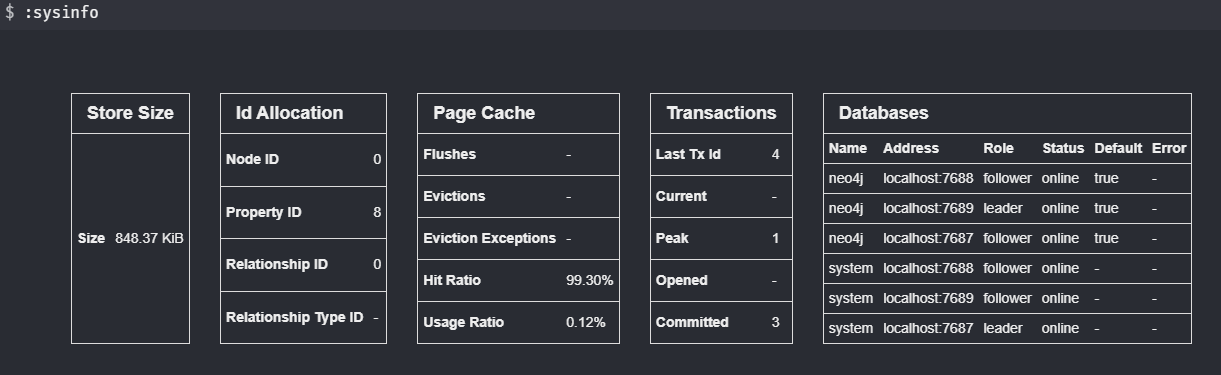
Tabela 2. Podešavanja drugog i trećeg servera jezgra.

Nakon podešavanja sva tri servera, klaster je spreman za rad. Pokretanjem drugog i trećeg servera, ispisuje se poruka u konzoli prvog, što se može videti na slici 9.



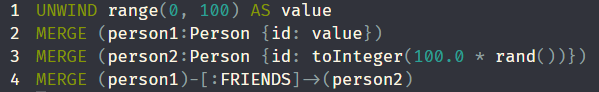
Slika 9. Poruka o uspešnom pokretanju kauzalnog klastera.

Za upravljanje klasterom, potrebno je povezati se na neki od servera. Kao što je navedeno u podešavanju, pristup prvom serveru obavlja se sa adrese **localhost:7474**, gde je potrebno ulogovati se sa unapred postavljenim korisničkim imenom i lozinkom **neo4j/neo4j** (koje treba promeniti kasnije). Izvršavanjem komande **:sysinfo** proverava se status klastera i ispisuju osnovne informacije, čiji je primer dat na slici 10. Neke od informacija uključuju poslednje identifikatore, informacije o kešu, poslednje stavke Raft zapisa kao i uloge u bazama podataka. U konkretnom slučaju, prvi server je vođa baze podataka **system**, dok je treći server vođa baze podataka **neo4j**. Ove dve baze su deo svake Neo4j instance.



Slika 10. Osnovne informacije o kauzalnom klasteru.

Provera rada kauzalnog klastera vrši se upisom podataka na jedan server (koji je vođa za bazu podataka), dok se pokušaj čitanja obavlja sa nekog drugog. Ukoliko klaster radi ispravno, podaci će biti dostupni na svim serverima. Na slici 11 prikazan je korišćeni zahtev za upisom (sa „dummy“ podacima), dok je poruka o uspešnom izvršenju prikazana na slici 12. Pokušaj upisa na nekom od pratilaca vraća grešku jer rutiranje zahteva nije omogućeno u osnovnom podešavanju.

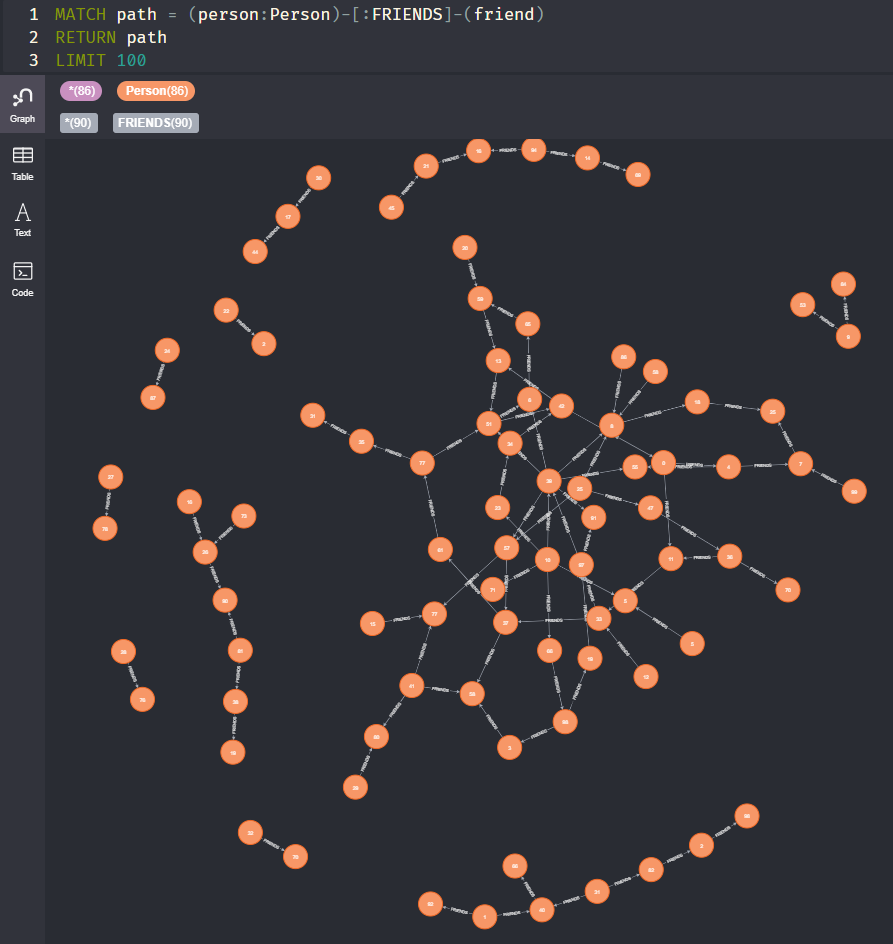


Slika 11. Ubacivanje „dummy“ podataka o prijateljima.



Slika 12. Poruka o uspešnom kreiranju čvorova i veza grafa.

Za proveru rada, moguće je konektovati se na neki od preostala dva servera. Pristupom lokaciji **localhost:7475,** vrši se povezivanje sa drugim serverom. Prikaz upita i rezultata pretrage grafa nalazi se na slici 13. Na osnovu dobijenih rezultata, može se zaključiti da klaster ispravno radi.



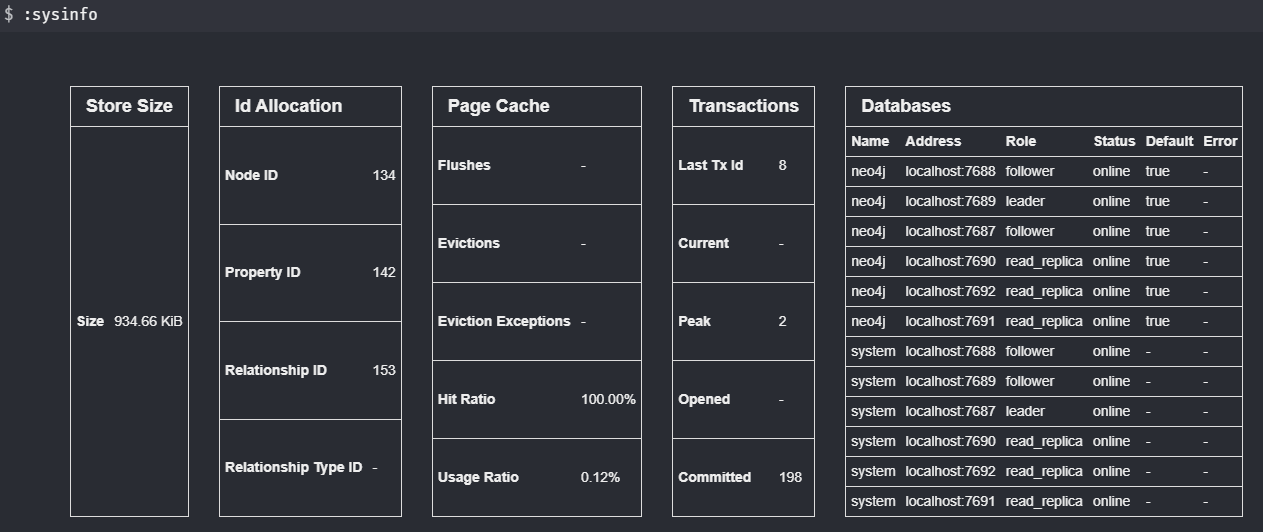
Slika 13. Uspešno izvršenje upita za čitanje.

Sledeći korak u proširenju klastera jeste dodavanje replika za čitanje. Podešavanje replika veoma podseća na podešavanje jezgra, uz neznatne izmene (mod rada, odsustvo Raft protokola). U tabeli 3, mogu se videti podešavanja sve tri replike.

|  |  |  |
| --- | --- | --- |
| Replika 1 | Replika 2 | Replika 3 |
| dbms.mode=READ\_REPLICA | | |
| causal\_clustering.initial\_discovery\_members=localhost:5000,localhost:5001,localhost:5002 | | |
| causal\_clustering.discovery\_listen\_address=:5003 | causal\_clustering.discovery\_listen\_address=:5004 | causal\_clustering.discovery\_listen\_address=:5005 |
| causal\_clustering.transaction\_listen\_address=:6003 | causal\_clustering.transaction\_listen\_address=:6004 | causal\_clustering.transaction\_listen\_address=:6005 |
| dbms.connector.bolt.listen\_address=:7690 | dbms.connector.bolt.listen\_address=:7691 | dbms.connector.bolt.listen\_address=:7692 |
| dbms.connector.http.listen\_address=:7477 | dbms.connector.http.listen\_address=:7478 | dbms.connector.http.listen\_address=:7479 |
| dbms.connector.https.listen\_address=:6477 | dbms.connector.https.listen\_address=:6478 | dbms.connector.https.listen\_address=:6479 |
| dbms.backup.listen\_address=0.0.0.0:6365 | dbms.backup.listen\_address=0.0.0.0:6366 | dbms.backup.listen\_address=0.0.0.0:6367 |

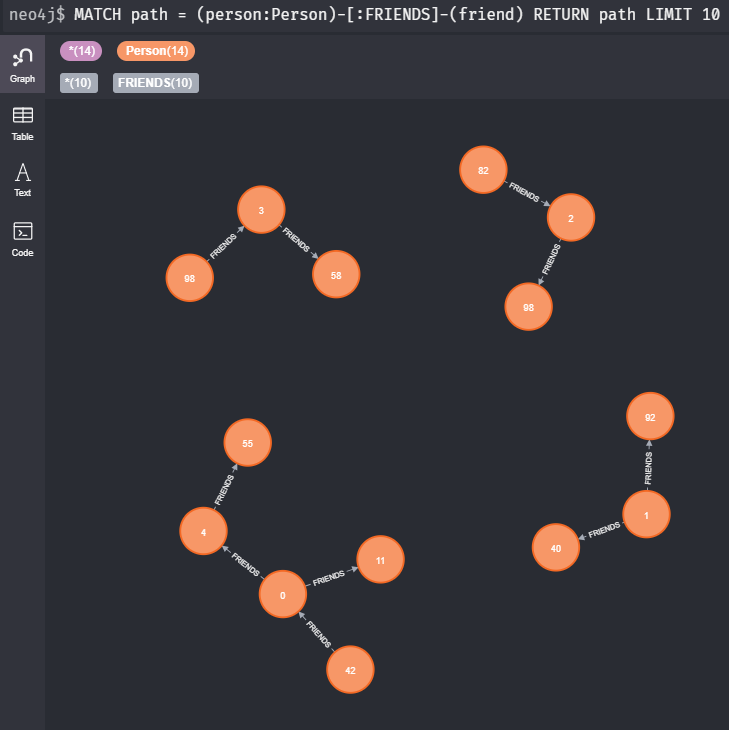
Tabela 3. Podešavanje replika za čitanje.

Pokretanje replika se vrši na potpuno isti način. Ponovnim pokretanjem **:sysinfo** komande, mogu se videti novonastale izmene nad klasterom, prikazane na slici 14. Uočava se nova uloga koju imaju replike (read\_replica).



Slika 14. Prikaz stanja klastera nakon dodavanja replika za čitanje.

Ukoliko su replike ispravno povezane, čitanje sa neke od njih bi trebalo da da iste rezultate kao i čitanje sa nekog od servera jezgra. Međutim, treba obratiti pažnju da se pri logovanju upotrebi **bolt://** šema (za servere jezgra je to bila **neo4j://** šema, koja je ujedno i osnovno podešavanje), jer u suprotnom pribavljanje podataka neće biti moguće. Rezultati uspešnog upita za čitanje, mogu se videti na slici 15.



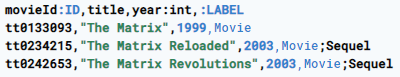
Slika 15. Uspešno čitanje sa replika za čitanje.

Sada je klaster potpuno funkcionalan i spreman za rad. U realnim situacijama, tri servera jezgra i tri replike često ne predstavljaju dovoljan nivo sigurnosti i propusnosti. Zbog toga je bilo kad u toku rada moguće dodati nove servere, a po potrebi i ukloniti ih (najčešće se uklanjaju replike, dok su serveri jezgra postojani). Naknadno dodavanje novih servera prati istu logiku – većina podešavanja ostaje ista, dok se uvećavaju samo brojevi portova. Naravno, ukoliko je to potrebno, konfiguracioni fajl sadrži pregršt opcija koje će koristiti kako početnicima, tako i iskusnijim korisnicima.

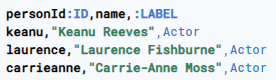
## Rad sa podacima

Često se upotreba klastera prirodno nagoveštava kada potrebni resursi prevaziđu mogućnosti jedne mašine. Prema tome, najčešći oblik rada sa podacima predstavlja „backup/restore“ funkcionalnost, tj. podaci se iz prethodne baze podataka kopiraju u novi klaster. Međutim, kako ova funkcionalnost predstavlja oblast za sebe (a takođe u konkretnom slučaju nema prethodnih podataka), akcenat će biti na korišćenju alata **neo4j-admin** i komande **import** za populaciju graf baze podacima.

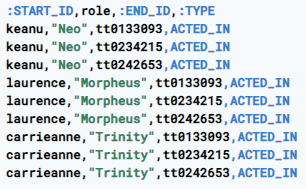
Za demonstraciju rada, biće korišćeni podaci iz Neo4j dokumentacije u obliku CSV datoteka, koji će biti unešeni u bazu podataka „test“. Potrebno je da se sve datoteke nalaze u posebnom poddirektorijumu **import** u okviru neo4j direktorijuma. Na slikama 16, 17 i 18 može se videti sadržaj datoteka o filmovima, glumcima i njihovim ulogama, respektivno. Treba napomenuti da se potpunija baza podataka može naći kao jedan od primera u okviru veb interfejsa Neo4j servera.



Slika 16. CSV datoteka o filmovima.

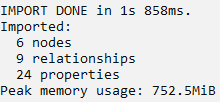


Slika 17. CSV datoteka o glumcima.



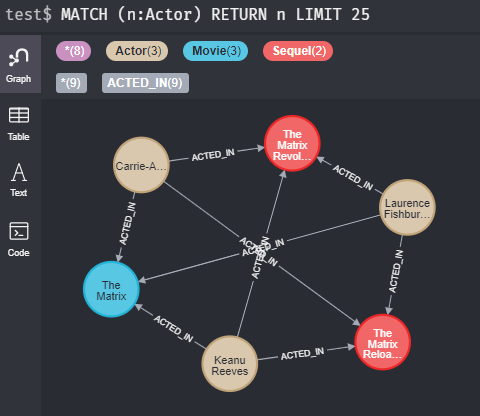
Slika 18. CSV datoteka o ulogama.

Pokretanjem komande: **neo4j-admin import --database=test --nodes=import/movies.csv --nodes=import/actors.csv --relationships=import/roles.csv**, dobija se obaveštenje o uspešnom dodavanju, što se može videti na slici 19. Opcijom **--nodes** dodaju se čvorovi, dok se opcijom **--relationships** dodaju potezi.



Slika 19. Poruka o uspešnom dodavanju čvorora i potega.

Ono što je važno istaći jeste da korišćenjem ove komande, podaci postoje samo na jednom serveru. Da bi i ostali serveri ispravno videli podatke, potrebno je da oni budu ručno učitani kombinacijom komandi **dump** i **load**. Korišćenjem prve komande, pravi se kopija baze sa podacima, a onda se drugom komandom ona učitava na svaki od servera (dovoljno na serverima jezgra). Podaci nakon dodavanja bivaju dostupni tek nakon restartovanja klastera. Preporučuje se dodavanje podataka kada klaster nije u radu, da ne bi došlo do potencijalnih konflikata sa podacima. Na slici 20 prikazana je baza podataka formirana prethodno prikazanim podacima.



Slika 20. Izgled test baze podataka.

Admin ubacivanje podataka podržava i dodatne opcije: korišćenje drugih CSV limitera, kombinovanje više fajlova odjednom, korišćenje istih labela, odn. imena veza, kao i preskakanje veza sa nepostojećim čvorovima ili istim identifikatorom.

# Zaključak

Upravljanje Neo4j klasterom predstavlja veoma širok pojam koja obuhvata veliki broj oblasti čiji opseg prevazilazi obim ovog rada. Zbog toga je akcenat bio na korišćenju klastera za potrebe malih aplikacija, koje zahtevaju svega nekoliko servera. Povećanjem složenosti aplikacija, povećava se i usložnjava struktura klastera. Brojni mehanizmi i protokoli (Raft, Bolt) postaju od većeg značaja sa postojanjem servera na većem broju lokacija, tj. pri korišćenju više centara za skladištenje podataka. Tada dolazi do značaja projektovanje i precizno raspoređivanje servera, kako bi se osigurala postojanost i funkcionalnost sistema. Loše projektovanje može dovesti do problema sa izborima, iako postoji dovoljan broj servera.

Kao još jedna značajna oblast u napredovanju funkcionalnosti klastera, ističe se Fabric koji predstavlja alat koji, između ostalog, omogućava formiranje virtuelnih baza podataka i particionisanje podataka, čime se rešava jedan od najvećih prepreka kad je skaliranje graf baza podataka u pitanju. Fabric je veoma mlada tehnologija koja se pojavila od verzije 4.0, međutim, već sada pruža mogućnosti koje podižu upotrebu klastera na viši nivo.

Od svog pojavljivanja 2007. godine, Neo4j baza podataka značajno je napredovala. Trenutno (jun 2021.), predstavlja najpopularniju graf bazu podataka po rangiranju sajta DB-Engines (<https://db-engines.com/en/ranking>), i nema sumnje da će tako i ostati u narednom periodu. Jednostavni i intuitivni način upotrebe, kako jedne instance, tako i klastera, predstavlja odličan izbor kako početnika, tako i naprednijih korisnika.

# Literatura

1. Knjiga - Learning Neo4j 3.x by Jerome Baton, Rik Van Bruggen
2. Knjiga - Neo4j in Action by Aleksa Vukotic, Nicki Watt, Tareq Abedrabbo, Dominic Fox, Jonas Partner
3. <https://neo4j.com/docs/operations-manual/current/clustering/> - sa podkategorijama
4. <https://neo4j.com/docs/operations-manual/current/tutorial/> - sa podkategorijama
5. <https://neo4j.com/docs/operations-manual/current/clustering-advanced/> - sa podkategorijama
6. Slike 1, 2, 3, 4, 5 preuzete sa Neo4j zvanične dokumentacije o klasterizaciji (literatura 3, 4 i 5)
7. Slike 6, 7 preuzete iz knjige (literatura 2)