TESTIRANJE PERFORMANSI VELIKIH BAZA PODATAKA U RADU SA VELIKIM KOLIČINAMA PODATAKA - PRIMJER CASSANDRA BAZE PODATAKA

TESTING THE PERFORMANCE OF LARGE DATABASES WHEN WORKING WITH LARGE AMOUNTS OF DATA - EXAMPLE CASSANDRA DATABASE

soft. ing. Edin Kohnić
Van. prof. dr. Denis Čeke
Van. prof. dr. Nevzudin Buzađija#
Univerzitet u Zenici, Bosna i Hercegovina

REZIME
U procesu prikupljanja i analize velikih količina podataka, pojavljuju se izazovi koji se odnose na postizanje optimizacije vremena potrebnog za dohvatanje željenih podataka iz jako velikih skupova podataka smještenih u različitim bazama podataka.Ovaj rad istražuje performanse distribuirane baze podataka Cassandra pri radu s velikim skupovima podataka (500k–1M zapisa). Kreirana je testna baza podataka, a performanse su analizirane kroz operacije čitanja i pisanja koristeći alate poput TRACING i CFStats. Testirani su različiti scenariji, uključujući uticaj strukture particionog ključa i veličine podataka na efikasnost upita. Rezultati pokazuju da optimizacija arhitekture i pravilno modeliranje podataka značajno poboljšavaju performanse Cassandre u skalabilnim sistemima.

Ključne riječi: Cassandra, veliki skupovi podataka, performanse, tracing, cfstats

ABSTRACT
In the process of collecting and analyzing large amounts of data, challenges arise related to optimizing the time required to retrieve desired data from very large data sets located in different databases. This paper investigates the performance of the distributed database Cassandra when working with large data sets (500k–1M records). A test database was created, and performance was analyzed through read and write operations using tools such as TRACING and CFStats. Various scenarios were tested, including the impact of partition key structure and data size on query efficiency. The results show that architectural optimization and proper data modeling significantly improve Cassandra performance in scalable systems.

Keywords: Cassandra, large datasets, performance, tracing, cfstats

1. UVOD

Distribuirane baze podataka, poput Cassandre [1], igraju ključnu ulogu u upravljanju velikim količinama podataka u skalabilnim sistemima. Cassandra je dizajnirana za horizontalnu skalabilnost, visoku dostupnost i otpornost na greške, što je čini pogodnom za aplikacije koje rade s velikim podacima. Cilj ovog rada je testirati performanse Cassandre kroz analizu operacija čitanja i pisanja nad velikim skupovima podataka, uz korištenje alata *TRACING* i

CFStats [2, 3]. Fokus ovog istraživanja je na utjecaju veličine podataka, strukture particionog ključa i različitih tipova upita na efikasnost sistema.

2. CASSANDRA – KLJUČNE KARAKTERISTIKE

Apache Cassandra je distribuirana NoSQL baza podataka optimizovana za rad s velikim količinama podataka i visokim zahtjevima za dostupnošću. Razvijena je za horizontalnu skalabilnost, gdje se podaci distribuišu po čvorovima bez centralnog kontrolera, što omogućava otpornost na greške i efikasno rukovanje velikim skupovima podataka [1] [4, 5, 6].
Ključne karakteristike Cassandre uključuju:

- Distribuirana arhitektura – svi čvorovi su ravnopravni, bez centralnog kontrolera.
- Linearno skaliranje – dodavanjem novih čvorova povećava se kapacitet baze.
- Visoka dostupnost – podaci se repliciraju na više čvorova, čime se osigurava otpornost na kvarove.
- Efikasne operacije čitanja i pisanja – koristi commit log, memtable i *SSTable* za optimizaciju performansi.
- Denormalizacija podataka – umjesto složenih *JOIN* upita, Cassandra koristi dizajn zasnovan na particionim ključevima kako bi upiti bili brži.

Ove karakteristike čine Cassandru pogodnom za aplikacije koje zahtijevaju skalabilnost i visoku dostupnost [7, 8].

3. HIPOTEZE ISTRAŽIVANJA

Cilj istraživanja je procijeniti performanse Cassandra baze podataka pri radu s velikim skupovima podataka, analizirajući različite scenarije upita i njihovu efikasnost. Formulisane su tri hipoteze:

- H_0: (nulta hipoteza): Povećanje broja zapisa (500k–1M) neće značajno uticati na vrijeme izvršenja upita, jer Cassandra efikasno upravlja velikim podacima.
- H_1: (alternativna hipoteza): Povećanje obima podataka dovodi do rasta vremena izvršenja upita zbog povećanog opterećenja sistema.
- H_2: Struktura particionog ključa direktno utiče na performanse upita – pravilno odabrani ključevi mogu optimizovati pretragu i smanjiti latenciju.

Testiranjem ovih hipoteza istražuju se faktori koji utiču na efikasnost Cassandre, kako bi se dala preporuka za optimizaciju performansi u radu s velikim podacima.

4. KREIRANJE RADNOG OKRUŽENJA

Za testiranje performansi Cassandre, postavljeno je eksperimentalno okruženje na jednom čvoru, koristeći lokalnu instalaciju Apache Cassandre. Pristup bazi ostvaren je putem *CQLSH* (Cassandra Query Language Shell), a podaci su generisani pomoću Python skripti i ubačeni u bazu koristeći batch operacije.

4.1. Struktura baze podataka

Kreirane su tri tabele:

- ***employees*** – sadrži podatke o zaposlenima (ime, prezime, pozicija, odjeljenje, datum zaposlenja).
- ***status_history*** – bilježi promjene statusa zaposlenih (employee_id, datum promjene, nova pozicija).
- ***salaries*** – sadrži podatke o platama (nivoi plata, datumi isplate, employee_id).

Podaci su strukturirani tako da optimizuju performanse upita, s fokusom na particione ključeve.

4.2. Generisanje i unos podataka

Za testiranje performansi, u tabele je ubačeno **800.000 zapisa** po tabeli, koristeći **Python Faker biblioteku** za generisanje realističnih podataka. Unos je realizovan pomoću **BatchStatement** operacija, čime je smanjen broj pojedinačnih INSERT upita i poboljšana efikasnost unosa podataka.

Ovo okruženje omogućilo je testiranje Cassandre pri radu s velikim količinama podataka i evaluaciju njenog ponašanja pod opterećenjem.

5. TESTIRANJE PERFORMANSI

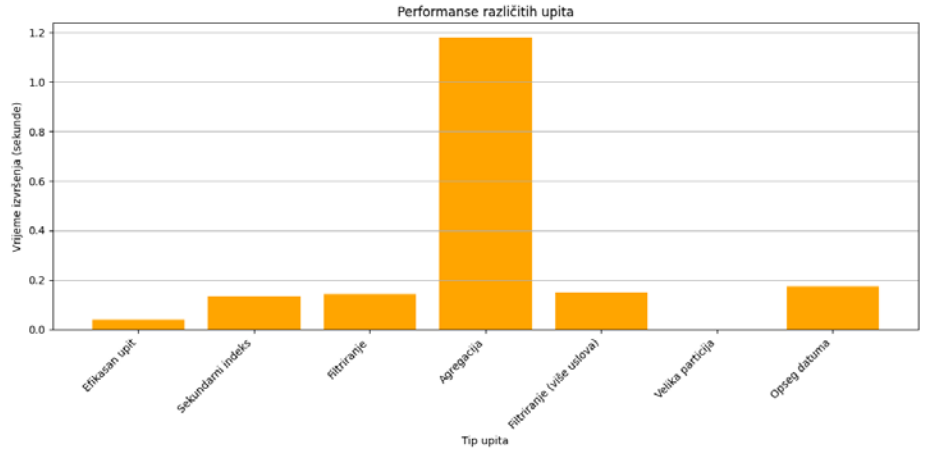
Cilj testiranja je analizirati efikasnost Cassandre pri čitanju i pisanju podataka koristeći različite vrste upita i veličine datasetova (500k–1M zapisa). Testovi su izvedeni pomoću Python skripti koje su mjerile vrijeme izvršenja upita.

5.1. Testiranje čitanja podataka

Izvedeni su različiti upiti kako bi se ispitao uticaj particionog ključa i ALLOW FILTERING opcije na performanse:

- Efikasan upit (particioni ključ) – najbrži rezultati (~0.04 sek).
- Upiti sa sekundarnim indeksom – sporiji (~0.13 sek).
- Neefikasni upiti sa ALLOW FILTERING – značajno sporiji (~0.17–1.18 sek).
- Upiti sa velikim particijama – mogu izazvati greške zbog prevelikog opterećenja.

Rezultati pokazuju da upiti koji koriste particioni ključ imaju značajno bolje performanse u odnosu na one koji koriste filtriranje ili agregacije.

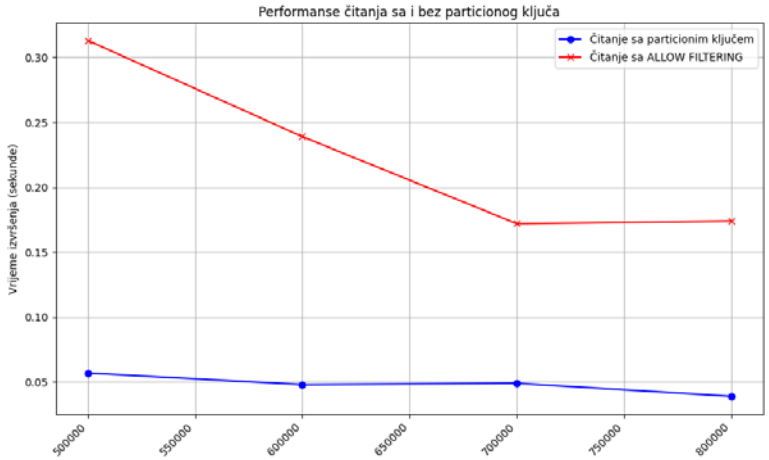


#

Slika 1. Rezultati izvršenja čitanja nad velikim skupom podataka za različite vrste upita

5.2. Performanse s različitim veličinama skupova podataka

U ovom dijelu je testirano kako broj zapisa utiče na vrijeme izvršenja upita:



Slika 2. Rezultati izvršenja čitanja nad različitim veličinama skupova podataka (500k do 800k) sa i bez korištenja particionog ključa i funkcionalnosti ALLOW FILERTING

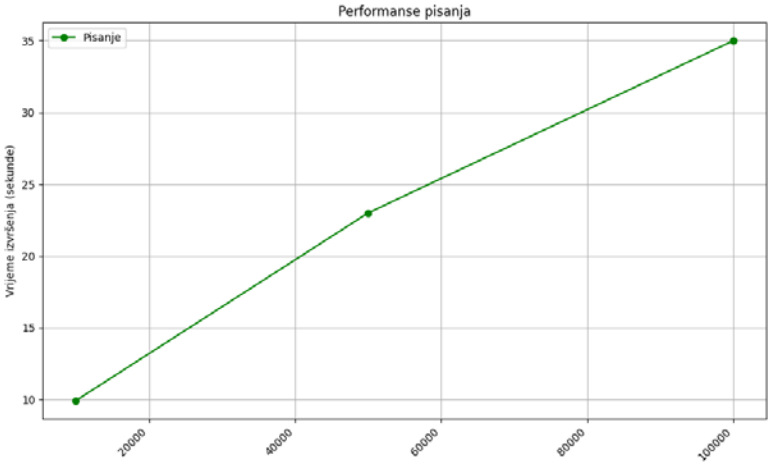
Analizom rezultata jasno je da upiti koji koriste particioni ključ omogućavaju efikasnu pretragu sa minimalnim vremenima izvršenja. Sa druge strane, upiti sa ALLOW FILTERING su manje efikasni i zahtijevaju više vremena za izvršenje, ali sa većim brojem podataka, performanse se povremeno poboljšavaju zbog optimizacije raspodjele podataka. Korištenje particionog ključa je ključno za postizanje optimalnih performansi u Cassandra bazi podataka, dok upiti sa ALLOW FILTERING treba da se koriste sa pažnjom, posebno kod većih skupova podataka, jer značajno povećavaju latenciju i vrijeme izvršenja.

5.3. Testiranje pisanja podataka

Unos podataka testiran je na 10k, 50k i 100k zapisa. Vrijeme unosa raste sa veličinom skupa podataka:

- 10k zapisa – ~9.9 sekundi
- 50k zapisa – ~22.9 sekundi
- 100k zapisa – ~35 sekundi

Unos podataka je skalabilan, ali pokazuje blagi rast latencije kako se povećava broj zapisa.



Slika 3. Rezultati izvršenja upisa različite veličine podataka u bazu

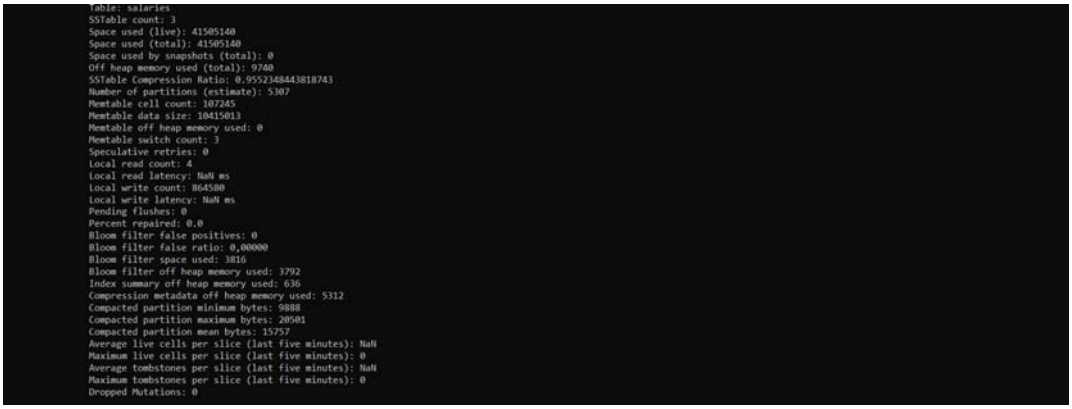
5.3. Analiza pomoću *CFStats* i *TRACING*

Analiza je pokazala da:

- *CFStats* potvrđuje optimizovano skladištenje podataka, ali povećanje *SSTable* fajlova može uticati na brzinu upita.
- *TRACING* analiza otkriva da složeni upiti zahtijevaju dodatno pretraživanje particija, što povećava latenciju.

Zaključak testiranja

Najbolje performanse postižu se upitima koji koriste particioni ključ i optimizovane strukture podataka. *ALLOW FILTERING* značajno povećava latenciju i treba ga izbjegavati gdje god je moguće.



Slika 4. Statistički podaci tabele salaries

Slika 5. Tracing ON – praćenje izvršenja upita bez korištenja ALLOW FILTERING – manji broj zapisa

6. REZULTATI ISTRAŽIVANJA I DISKUSIJA

Na osnovu sprovedenih eksperimenata testirane su postavljene hipoteze:

- H_0 (nulta hipoteza): Povećanje broja zapisa nije značajno uticalo na performanse upita koji koriste particioni ključ. Vrijeme izvršenja ovih upita ostalo je stabilno čak i pri 800k zapisa (~0.0340–0.0560 sek).
- H_1 (alternativna hipoteza): Povećanje obima podataka imalo je značajan uticaj na sporije upite, posebno one sa ALLOW FILTERING. Vrijeme izvršenja ovih upita variralo je od 0.3130 sek (500k zapisa) do 0.1740 sek (800k zapisa), što ukazuje na nepredvidive performanse kod filtriranja bez particionog ključa.

- H_2 (uticaj particionog ključa): Upiti koji koriste pravilno dizajnirane particione ključeve imali su najstabilnije performanse. Cassandra je bila znatno efikasnija kada su podaci raspoređeni po manjim particijama, dok su upiti sa velikim particijama ili loše definisanim ključevima rezultirali greškama ili sporim izvršenjem.

Ovi rezultati potvrđuju važnost optimizacije modela podataka i pravilnog dizajna particionih ključeva kako bi se postigle najbolje performanse.

7. ZAKLJUČAK

Istraživanje je pokazalo da Cassandra nudi visoku efikasnost i skalabilnost pri radu sa velikim količinama podataka, ali pod uslovom pravilne konfiguracije. Upiti koji koriste particioni ključ ostvaruju najbolje performanse, dok upiti sa ALLOW FILTERING mogu značajno povećati latenciju i treba ih izbjegavati gdje god je to moguće.
Najvažniji zaključci su:

- Cassandra je skalabilna i efikasna, ali dizajn particionog ključa direktno utiče na brzinu upita.
- Upiti koji koriste particioni ključ su i do 10x brži u odnosu na one koji koriste ALLOW FILTERING.
- Povećanje veličine baze podataka (500k–800k zapisa) nije negativno uticalo na efikasne upite, ali je imalo značajan uticaj na loše optimizovane upite.

Ovi nalazi mogu pomoći u optimizaciji rada s Cassandrom u velikim bazama podataka, naglašavajući važnost pravilnog modelovanja podataka i optimizacije upita.

8. REFERENCE

[1] Cassandra Basics, [Online], Dostupno: https://cassandra.apache.org/_/cassandra-basics.html, [Pristupano: 13-April-2025]

[2] What Is Cassandra? Meaning, Working, Features, and Uses, [Online], Dostupno: https://www.spiceworks.com/tech/big-data/articles/what-is-cassandra/, [Pristupano: 21-Mart-2025

[3] Cassandra Data Model Definition, [Online], Dostupno: https://www.scylladb.com/glossary/cassandra-data-model/, [Pristupano: 04-April-2025]

[4] A Lakshman, P.Malik, Cassandra - A Decentralized Structured Storage System, ACM SIGOPS Operating Systems Review, Vol(44), Issue 2, pp 35-40, 2010, https://doi.org/10.1145/1773912.1773922

[5] S. Katam, Performance Tuning of Big Data Platform Cassandra Case Study, Faculty of Computing, Blekinge Institute of Technology, Karlskrona, Sweden, Computer Science, Engineering, 2016, https://api.semanticscholar.org/CorpusID:57960663

[6] Apache Cassandra Architecture Inside DataStax Distribution of Apache Cassandra, [Online] Dostupno: https://apexassembly.com/wp-content/uploads/2020/08/DataStax-WP-Apache-Cassandra-Architecture_Technical_4_2.pdf, [Pristupano: 02-Dec-2024]].

[7] "The Apache Cassandra Project.", [Online], Dostupno: http://cassandra.apache.org/. [Pristupano: 03-Dec-2024]

[8] Charsyam - Cassandra Data Model [Online], Dostupno: https://charsyam. wordpress.com/tag/cassandra-data-model/ [Pristupano: 07-Mart-2025]].