Napredni algoritmi i strukture podataka

Predmetni projekat

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Projekat

- Timski rad - timovi od 3-5 studenta
- Ocenjivanje će se vršiti za svakog člana tima **posebno**!
- Obavezna upotreba sistema za kontrolu verzija (Git)
- Obavezna upotreba sistema za praćenje projekta (GitHub)

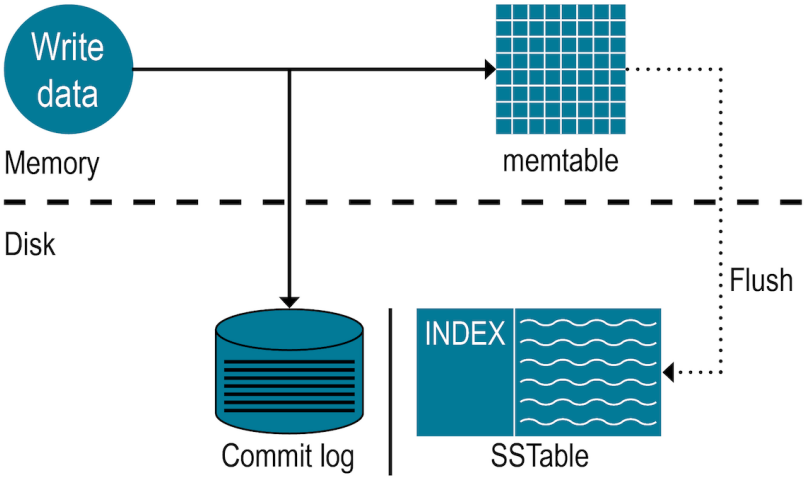
Projektni zadatak

- Key-Value engine-a za skladištenje podataka
- Ograničenje broja zahteva korisnika
- Keširanje sadržaja
- Implementacija koristeći programski jezik Golang
- Implementacija koristeći programski jezik C/C++ (rad na svoju ruku)

Uvod

- Vaš zadatak je da implementirate Key-Value engine-a za skladištenje podataka
- Komponente koje trebate da imate u projektu su:
  1. Ispravan **Write path**
  2. Ispravn **Read path**
- Minimalne operacije koje sistem treba da podrži su:
  1. **PUT** — dodavanje podataka (prihvata ključ tipa **string**, i vrednost tipa **bit array**, i vraća vrednost tipa **bool**)
  2. **GET** — traženje inforamcija, (prihvata ključ tipa **string**, i vraća vrednost tipa **bit array**)
  3. **DELETE** — brisanje zapisa (prihvata ključ tipa **string**, i vraća vrednost tipa **bool**)

Write path

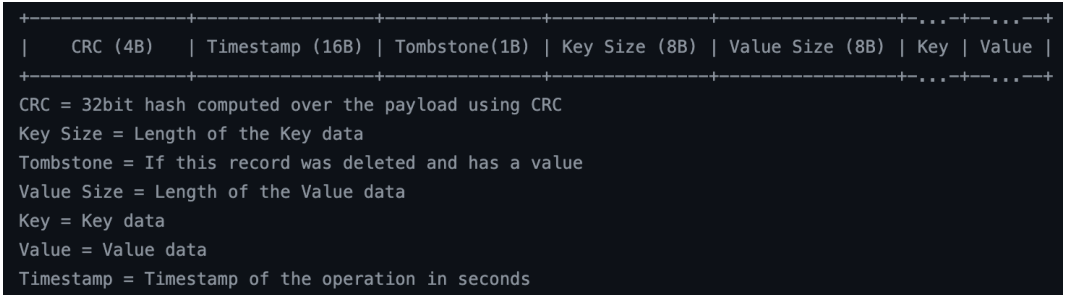

Write data
Memory
memtable
Disk
Flush
INDEX
Commit log
SSTable

1. Kada korisnik pošalje **PUT** zahtev vašem sistemu, zahtev se prvo zapisuje u Write Ahead Log (Commit Log) — WAL
2. Kada WAL potvrdi zapis, potrebno je da se podatak doda u Memtable, koji se nalazi **striktno** u memoriji
3. Kada se dostigne unapred definisana veličina Memtable-a, vrednosti se **sortiraju po ključu**, i formira se SSTable koji se zapisuje na disk
4. WAL implementirati kao segmentirani log
5. Veličinu segmenta, kao i količinu zapisa u Memtable je moguće podešavati kroz spoljni konfiguracioni fajl

- Prilikom formiranja SSTable-a, potrebno je ispravno kreirati sledeće elemente:
  - Data file — sadrži konkretne podatke
  - Filter (Bloom Filter) svih ključeva koji će biti sadržanu u Data fajlu
  - Index svih ključeva koji će biti sadržanu u Data fajlu
  - Summuary svih ključeva koji će biti sadržanu u Index fajlu
  - Metadata, Merkle stablo svih vrednosti iz Data fajla
- Prilikom formiranja fajlova, u nazivu je potrebno da se nalazi i nivo LSM Stabla za svaki element
- Dopušten broj nivao LSM stabla treba da se specificira kroz konfiguracioni fajl

WAL i Data struktura

Struktura WAL segmenata, i struktura Data fajla treba da bude kao na slici:



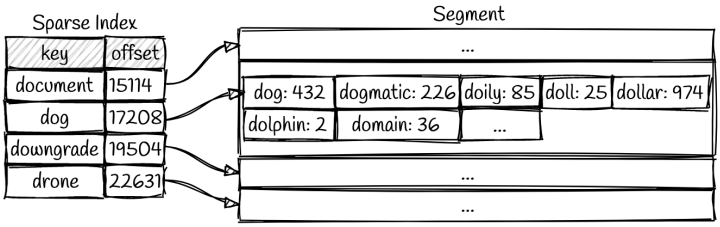
- Na timovima je ostavljeno da se odluče da li će koristiti identičnu strukturu za WAL i Data segment
- Moguće je koristiti potpuno identične strukture, ili ukloniti segmente koje smatrate da su višak
- Šta god da je odluka, trebaćite, potrebno je obrazložiti žasto ste doneli takvu odluku

Memtable struktura

- Memtable treba da se sadrži od strutkure podatka koja čuva konkretne podatke (videti detalje kod ocenjivanja)
- **DELETE** treba da bude implementirarno kao **logičko** brisanje, zato što nismo sigurni da li je taj podataka pristuan u nekom drugom SSTable fajlu — Postaviti **Tombstone** vrednost na **true** za odgovarajući ključ
- Izmene implementirati kao **in-place AKO** je takav ključ prisutan u stuktuturi. Ako nije, implementirati kao dodavanje novog elementa — u oba slučaja korisnik vrši poziv **PUT** operacije

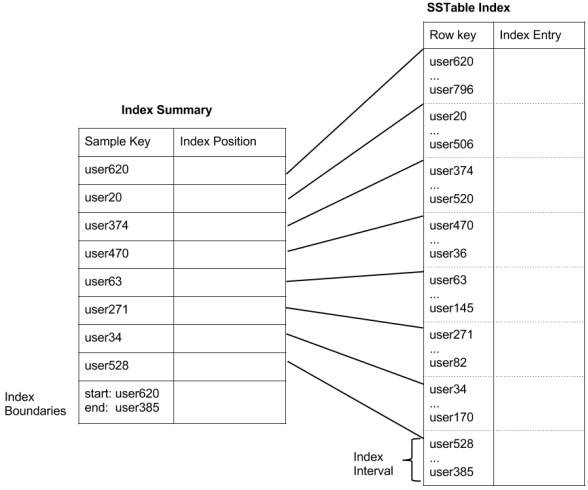
Index struktura

- Struktura Index-a, treba da bude kao na slici
- Sadrži niz elemenata, gde je prvi element **ključ** iz Data fajla, a druga vrednost je **pozicija** — offset ključa u Data fajlu

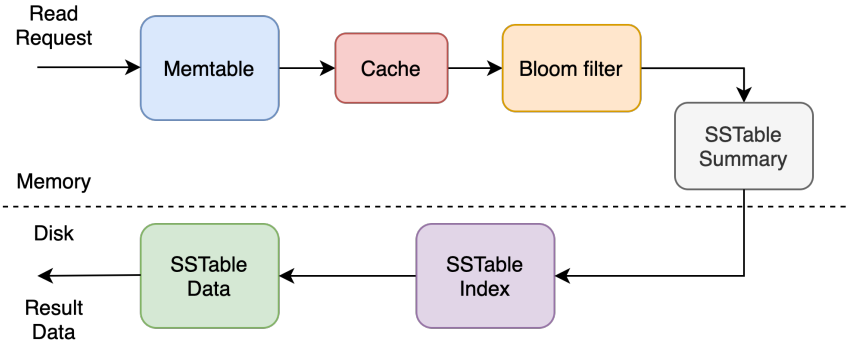


Summary struktura

- Struktura Summary-a, treba da bude kao na slici
- Na početku treba da sadži granice index fajla
- Dalje sadrži niz elemenata, gde je prvi element **ključ** iz Index fajla, a druga vrednost je **pozicija** ključa u Index fajlu



Read path


Read
Request
Memtable
Cache
Bloom filter
SSTable
Summary
Memory
Disk
SSTable
Data
SSTable
Index
Result
Data

- Kada korisnik uputi **GET** zahtev moramo da izvršimo sledeći put:
  - Prvo proveriti da li je ključ već prisutan unutar **Cache** strukture, ako jeste vratiti korisniku odgovor
  - Ako nije, proveriti da li je ključ **možda** prisutan unutar **Bloom Filter-a**, i ako nije vratiti korisniku odgovor
  - Ako je možda prisutan, proveriti da li **Summary** opseg čuva taj ključ, ako je ne vratiti korisniku odgovor
  - Ako čuva vrednost, uzeti **poziciju** unutar Index fajla, i pozicinirati se na tu poziciju unutar **Index** fajla
  - Kada dobijemo **poziciju** iz **Index** fajla, pozicionirati se na **Data** deo i pročitati vrednost
  - Kada smo našli vrednost, prvo je dodati u **Cache**, zatim vratiti korisniku odgovor
  - **Samo Bloom Filter strukturu je potrebno striktno učitavati u memoriju**
  - **Summary strukturu** učitati u memoriju **ako** je ključ u njenom opsegu
  - **Za ostale strukture koristiti Seek operaciju**

- Povratna vrednost GET operacije može biti **bit array** ili **konkretan tip**
- Ako se vraća konkretan tip, onda se mora čuvati i informacija koji tip je zapisan
- Odluka je ostavljena na timovima da odluče
- Šta god odlučite, potrebno je obrazložiti odluku

Cache stuktura

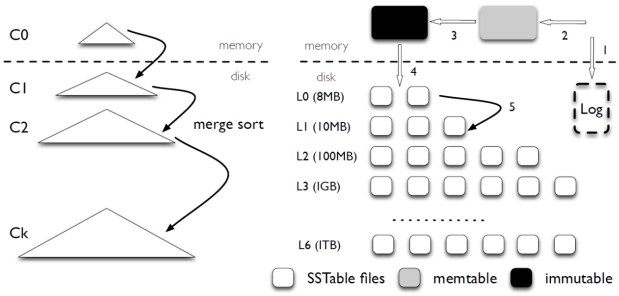
- Cache implementirati koristeći **LRU** algoritam
- Broj elemenata koliko će cache čuvati omogućiti da se čita iz konfiguracionog fajla
- Cache treba da se nalazi u memoriji

Konfiguracije

- Struktura konfiguracionog fajla, je ostavljena timu na izbor
- Od struktura moěte koristiti bilo šta (JSON, XML, YAML, TXT datotkea, ...)
- Obezbediti da postoji podrazumevana (**default**) konfiguracija implementirana kroz programski kod, ako eksterna konfiguracija nije zadata

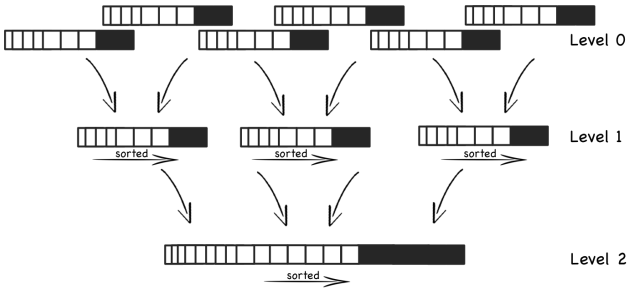
LSM Stabla

- Projekat organizovati kroz LSM stabla kao na slici
- Prvi nivo C_0 Memtable, a drugi nivo C_1 SStable
- Broj nivoa ograničiti kroz konfiguracioni fajl (videti detalje kod ocenjivanja)



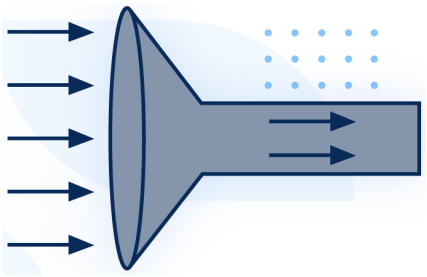
Kompakcije

- Omogućiti kompkacije viňižih nivao u više nivoe do nivao C_K, specificiranog kroz konfiguracioni fajl (videti detalje kod ocenjivanja)
- Kompakcije pokrenuti **ručno**
- Voditi računa o **Tombstone** elementima i ispravnom **merge sort-u**



Ograničenje stope/brzine pristupa

- Ograničenje stope/brzine pristupa vašem sistemu implementirati koristeći **Token Bucket** algoritam (videti detalje kod ocenjivanja)
- Sve potrebne parametre za ovaj algoritam čuvati u **vašem sistemu**
- Ovaj deo je obraničenje sistema — **štiti sam sebe**
- Podešenja čuvati u konfiguracionom fajlu



Ocena 6

- Za ocenu **6**, studenti treba da implementiraju **write path** i **read path** u potpunosti
- Za strukturu Memtable-a, **treba** da koriste **hash mape (map tup u golangu)**
- Ne treba da implementiraju eksterne konfiguracije, dovoljno je da imaju **default konfiguracije** u programskom kodu
- Ne treba implementirati Summary, LSM Stabla — Kompakcije, Ograničenje stope/brzine pristupa ni keširanje sadržaja, kao ni Merkle stabla
- Index fajl možete učitati u memoriju u potpunosti

Ocena 7

- Za ocenu **7**, studenti treba da implementiraju **write path** i **read path** u potpunosti
- Za strukturu Memtable-a, **treba** da koriste **SkipList** strukturu
- Ne treba da implementiraju eksterne konfiguracije, dovoljno je da imaju **default konfiguracije** u programskom kodu
- Ne treba implementirati LSM Stabla — Kompakcije, Ograničenje stope/brzine pristupa ni keširanje sadržaja, kao ni Merkle stabla
- Summary fajl možete učitati u memoriju u potpunosti

Ocena 8

- Za ocenu **8**, studenti treba da implementiraju, sve funkcionalnosti za ocenu 7
- Dodatno je potrebno da se implementira keširanje sadržaja
- Dodatno je potrebno da se implementira provera da li je ključ u opsegu Summary-a, bez učitavanje u memoriju

Ocena 9

- Za ocenu **9**, studenti treba da implementiraju, sve funkcionalnosti za ocenu 8
- Dodatno je potrebno da se implementira i Ograničenje stope/brzine pristupa
- Dodatno je potrebno da se implementira i Merkle stablo podataka, koje će biti sačuvano u **Metadata.txt** fajl

Ocena 10

- Za ocenu **10**, studenti treba da implementiraju, sve funkcionalnosti za ocenu 9
- Dodatno je potrebno da se implementira kompakcija sdržaja koja se pokreće ručno
- Minimalan broj LSM stabla je 4:
  - C_0 Memtable
  - C_1 Serijalizovana Memtable u SSTable
  - C_2 spojene dve SSTable nivao C_1
  - C_3 spojene dve SSTable nivao C_2
- Voditi računa da kada imate više nivao LSM stabla (više od 3), kompakcije moraju da se ispravno dese na svim nivoima

Ocena 10+ — gratis usmeni

- Za ocenu **10+**, studenti treba da implementiraju, sve funkcionalnosti za ocenu 10
- Dodatno je potrebno da se omogući da postoje tipovi **HyperLogLog — HLL**, **Count-min-sketch — CMS**
- Ove tipove možemo čuvati pod specifičnim ključem, serijalizovano kao niz bajtova
- Voditi računa da će verovatno morati da se doda nova operacija koja proširuje standardan **PUT**

Pitanja

Pitanja :) ?