Napredni algoritmi i strukture podataka

SSTable, Index, Formiranje

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

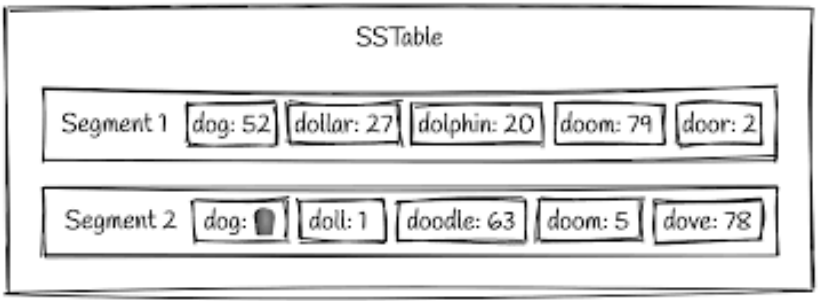
SStable - ideja

- Ideja iza tabele sortiranih stringova (**SSTable**) je relativno jednostavna
- To je niz parova **ključ-vrednost** koji su sortirani, i zapisani na disk
- **SStable** je nepromenljiva struktura (log based struktura) — nema brisanja ni izmene sadržaja
- Memtable **sortirati** pa zapisati na disk, sa relativno sličnom strukturom (ako ne i istom)
- Pod nizom sortiranih stringova ne misli se doslovno na stringove, već na **niz bajtova**
- Ključ će biti **string** to svkako, ali vrednost može biti bilo šta
- Zbog toga je najjednostavnije da se čuva niz bajtova

SStable — struktura

- Izabraćemo opšti oblik SSTable-a
- Niz parova **ključ-vrednost**, gde su obe vrednosti zapravo niz bajtova
- Pre zapisa Memtabe-a, sve vrednosti se **sortiraju** — ovo nam trebati kasnije!!
- Ovim smo dobili **data** segment
- Kada se obriše podatak, on nije momentalno uklonjen sa diska
- Sistem zapisuje specijalan podatak *Tombstone* da je neki ključ obrisan — markira ga za brisanje
- Brisanje elemenata je zapravo **nov zapis** u SStable — SSTable je nepromenljiva struktura

Opšta struktura SSTable



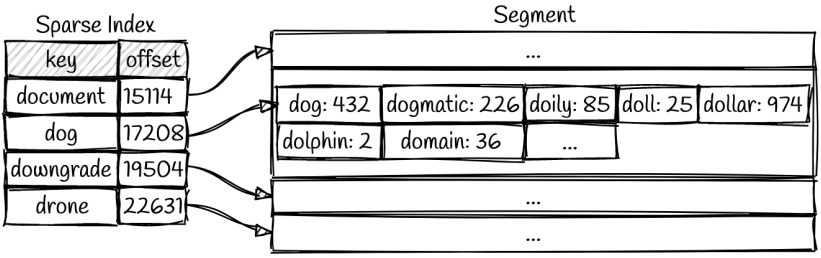
(LSM — Write Heavy Databases)

- Vidimo da je SSTable podeljen na nekakve blokove podataka
- Ono što nas za sada najviše zanima je struktura **data block** dela
- Ostali podaci nam nisu toliko zabavni **za sada** — videćemo ih kasnije **Data block** sadrži konkretne podatke u parovima **ključ-vrednost**, CRC proverom, vremenskom odrednicom, ...
- **ALI** podaci su kompresovani, mi se time neće baviti — ako nekoga zanima može da pogleda kako se podaci mogu kompresovati zarad uštede prostora
- **ALI** ova strukutra nam je već donekle poznata — hajde da svedemo na nešto što već znamo

```
+---------------+-----------------+---------------+---------------+-----------------+-...-+--...--+
|    CRC (4B)   | Timestamp (16B) | Tombstone(1B) | Key Size (8B) | Value Size (8B) | Key | Value |
+---------------+-----------------+---------------+---------------+-----------------+-...-+--...--+
CRC = 32bit hash computed over the payload using CRC
Key Size = Length of the Key data
Tombstone = If this record was deleted and has a value
Value Size = Length of the Value data
Key = Key data
Value = Value data
Timestamp = Timestamp of the operation in seconds
```

SSTable — Index

- Ideja iza svakog index-a je da što je pre moguće stignete do konkretnog sadržaja
- Pogledajte index pojmova u (svakoj) knjizi
- Ista ideja se koristi i kod SSTable-a
- Struktura je relativno jednostavna
- Sastoji se od dve vrednosti:
  1. **ključ** koji se nalazi u fajlu na disku
  2. **offset** u fajlu, tj. pozicija u fajlu na disku
- Fajl na disku u ovom slučaju je SStable!
- Index bi bilo lepo sortirati (ne budite lenji, učinite sebi uslugu :))



(LSM — Write Heavy Databases)

SSTable struktura

- Za primer, možemo da vidmo postojeći sistem za skladištenje podataka — Cassanda
- Ovaj sistem čuva svoje podatke na jednom mestu (kao i WAL)
- Na slici pored, vidimo da imamo dosta više delova od samo index-a i data dela :O
- **usertable-data-ic-1-TOC.txt** fajl sadrži spisak svih fajlova za konkretan SSTable

usertable-data-ic-1-CompressionInfo.db
usertable-data-ic-1-Data.db
usertable-data-ic-1-Filter.db
usertable-data-ic-1-Index.db
usertable-data-ic-1-Statistics.db
usertable-data-ic-1-Summary.db
usertable-data-ic-1-TOC.txt

(Distributed Datastore, SSTable format)

Filter

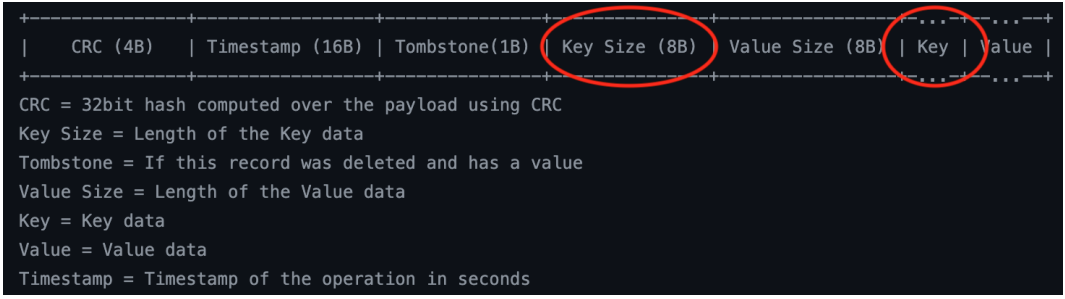
- **Filter** je zapravo zapisan **Bloom filter** na disk za nekakv skup ključeva
- Bloom filter se pita, **PRE** bilo kakvog indexa, da li se traženi ključ tu **ne** nalazi ili se **možda** nalazi
- Ako se **ne** nalazi, nema potrebe da otvaramo išta, možemo da gledamo dalje
- Ako je ključ **možda** tu, onda moramo proveriti, i za provere koristimo index strukture!!
- Njih koristimo da brže stignemo do podatka, **AKO** je ključ tu
- Bloom Filter-u trebamo da kažemo koliko elemenata se očekuje, ali to nije sada problem
- Ovaj podatak nam je unapred poznat, pošto tačno znamo koliko elemenata čuva Memtable
- Samim tim i za Bloom Filter imamo tu informaciju unapred poznatu!

Formiranje SSTabele

- Kada se Memtable napuni, ona se zapisuje na disk i formira SSTable
- Koristeći dostupne podatke dodatno formiramo;
  1. Bloom Filter za dostupnim kjučevima
  2. SSTable Index pošto znamo na kojoj poziciji u SSTable fajlu je koji ključ — sortirati
  3. TOC fajl u kom kažemo šta je sve od podatka dostpuno — svi prethodni elementi
- Potrebno je da formiramo **Merkle stablo** od podataka iz SSTable-a
- Nakon što je stablo formirano, stablo zapižemo u Metadata fajl

Potencijalno smanjenje prostora?

Šta sa ovim podacima...Da li treba da ih čuvamo kada već imamo Index...?



Zadaci

- Kada se Memtable (implementirana na prošlim vežbama) popuni do unapred definisanog praga, podatke sortirati i sačuvati na disk stvarajući SSTable
- Prilikom zapisivanja na disk, potrebno je ispravno kreirati sve definisane elemente koji čine SSTable (Data, Index, Filter, TOC, Metadata)
- Prilikom stvaranja elemenata za SSTable koristiti format imenovanja *usertable-GENERATION-ELEMENT.db*, gde
  - *ELEMENT* može biti nešto od Data, Index, Filter
  - *GENERATION* je indeksni broj koji se povecćava svaki put kada se kreira nova SSTtable-a
- Za Filter koristiti Bloom Filter implementiran na nekom od prethodnih vežbi
- Za Metadata koristiti implementaciju Merkle stabla implementiranog na nekom od prethodnih vežbi
- Elemente SSTable-a, možete kreirati u redosledu po izboru