Napredni algoritmi i strukture podataka

Segmentirani log, Brisanje delova loga, Izmene sa više klijenata

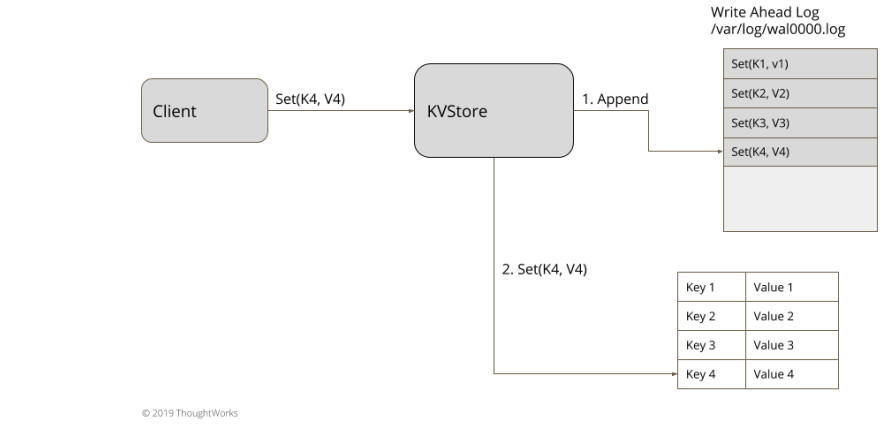
Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Kratka rekapitulacija

- WAL deluje kao rezervna kopija na disku za memorijsku strukturu tako što vodi evidenciju o svim operacijama nad tom strukturom
- U slučaju ponovnog pokretanja, memoriska struktura se može u potpunosti rekonstruisati ponavljanjem operacija iz WAL-a
- WAL koristi isključivo sekvencijalne I/O operacije za skladištenje podataka na disku
- WAL kao struktura podatka, direktno se oslanja na strukturu zasnovanu na log-u

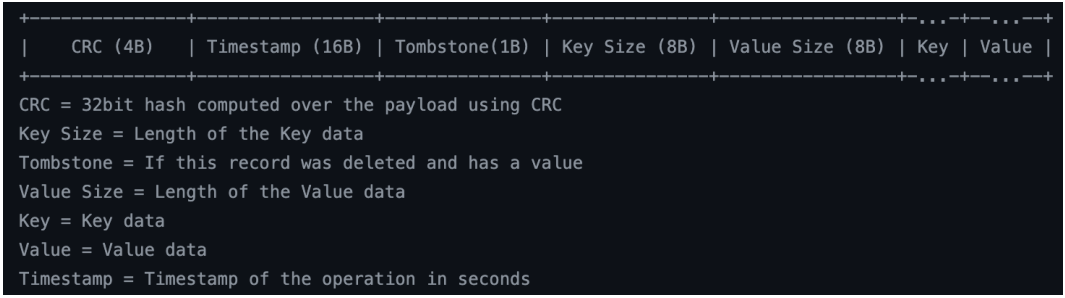
- Podatke u WAL možemo da dodamo, izmena nije dopuštena kao ni brisanje
- Izmena ili brisanje nekog podatka rezultuje novim zapisom u WAL
- WAL je *append-only* struktura, podaci se uvek dodaju na kraj
- WAL koristi mehanizam za proveru da li su zapisi ispravni prilikom učitavanja u memoriju

- WAL koristi baferisani I/O, da bi sprecio veliki broj operacija ka disku
- Za ovo se mogu koristiti sistemski pozivi — mmap
- WAL može da čuva informacije o transkacijama koje su se desile
- Ovo je bitno, zato što za svaku trankaciju, sve operacije se **moraju** izvršiti ili se transkacija odbacuje



(Martin Fowler Write-Ahead Log https://martinfowler.com/articles/patterns-of-distributed-systems/wal.html)

Format koji ćemo mi koristiti biće sličan RocksDB-u, ali malo uprošćen zbog jednostavnosti rada i naših potreba



- WAL zapisuje svoj sadržaj na hard disk
- Moguće je da se desi oštećenje zapisa
- CRC je jedan standardan način da nekako označimo sadržaj prilikom zapisa u WAL — *checksum*
- Proveriti jedinstvenost zapisa na promene prilikom čitanja podatka
- Ako oznaka nije ista, naš zapis nije više validan — došlo je do problema
- Ako jeste, možemo nastaviti sa radom
- Postupak ponoviti za svaki zapis unutar WAL-a

Write Ahead Log - Problem 1

Implementirali ste WAL mehanizam, ali ste ustanovili da on kreće da raste nekontrolisano, i kao takav počinje da bude težak za ikakvu obradu — pojedinačan WAL file počinje da bude usko grlo sistema.

Možemo li ovo poboljšati, ideje :) ?

Segmentirani Log

- WAL radi svoj posao prilično lepo — radi ono što mu je namena i ništa više
- To dovodi do problema — jedna datoteka WAL-a može da raste nekontrolisano i postane usko grlo, pri pokretanju
- Možemo brisati starije informacije, to je jedno rešenje
- Starije operacije nam više nisu potrebne
- **ALI** ta operacija čišćenja na jednoj ogromnoj datoteci je teško implementirati
- Može blokirati sistem (npr. GC u Javi)

- Možemo da nekako probamo da podelimo WAL datoteku
- Jedan WAL možemo podeliti u više segmenata
- Za svaki segment možemo da specificiramo veličinu
- Datoteke segmenata se skladište na disk nakon nekog definisanog praga
- U memoriji možemo da čuvamo samo zadnji segment
- Ovo su neke od standardnih ideja kako da rešimo problem — *divide and conquer* princip

Na primer RocksDB (Netflix), koristi veličinu zapisa, tj. svaki segment je iste veličine

```
+----------+----------+----------+----------------+--- ... ---+
| CRC (4B) | Size (2B) | Type (1B) | Log number (4B)| Payload   |
+----------+----------+----------+----------------+--- ... ---+
CRC = 32bit hash computed over the payload using CRC
Size = Length of the payload data
Type = Type of record
       (kZeroType, kFullType, kFirstType, kLastType, kMiddleType )
       The type is used to group a bunch of records together to represent
       blocks that are larger than kBlockSize
Payload = Byte stream as long as specified by the payload size
Log number = 32bit log file number, so that we can distinguish between
records written by the most recent log writer vs a previous one.
```

Jedan WAL možemo podeliti u više segmenata...

Gde da čuvamo segmente, ideje :) ?

Segmentacija loga - lokacija segmenata

- Segmente možemo da čuvamo na različitim mestima — nije baš praktično
- Segmente možemo da čuvamo na jednom mestu – može biti praktično
- Ako se odlučimo za drugu opciju, naš WAL treba da zna **samo** putanju do direktorijuma koji čuva segmente
- Ovaj direktorijum obično nosi naziv *wal* — gle čuda :)
- Ova ideja se može pokazati korisnom malo kasnije...

- Kada se WAL pokrene, on treba da preskenira *wal* direktrijum, i da pokupi lokacije segmenata
- Segmenti neće biti učitani u memoriju odmah
- Iz praktičnih razloga možemo da učitamo samo poslednji segment
- Ideja iza ovoga jeste brži pristup — pretpostavka: možda su podaci koje tražimo relativno skoro zabeleženi (imamo sreće)

Kada se WAL pokrene, on treba da preskenira *wal* direktrijum, i da pokupi lokacije segmenata. Segmenti neće biti učitani u memoriju odmah...

Pa šta onda da učitamo, ideje :) ?

- Postoji nekoliko strategija
- Jedna opcija je da pokupimo lokacije (putanje) do segmenata
- Ovo radimo da bi mogli lakše da im pristupimo — nema potrebe da ponovo skeniramo *wal* direktorijum
- **ALI** vvek treba da znamo redosled segmenata!
- Ovo nam pomaže da znamo šta smo pročitali kada radimo pretragu

Uvek treba da znamo redosled segmenata...

Kako ovo postići, ideje :) ?

Segmentacija loga - imenovanje segmenata

- Segmentacijom WAL-a moramo obezbediti jednostavan način za mapiranje offset-a WAL-a (ili rednih brojeva) u segmente
- Ovo se može uraditi na nekoliko načina
- Na primer, ime svakog segmenta se dobija spajanjem unapred poznatog prefiksa (npr wal) i offeta (ili rednog broja segmenta) — npr: wal_0001.log
- Ako je potrebno da pročitamo neki segment, moramo ga naći po identifikatoru na disku, i pretražiti sapise u njemu

Učitavanje segmenata

- WAL obično ima specijalizovan folder koji čuva sve segmente
- Kada se WAL pokreće, potrebno je da preskeniramo taj folder i da vidimo koliko segmenata ima
- Nakon toga, možemo da formiramo strukturu unutar WAL-a sa putanjama i offset-ima za vaki segmet, radi lakšeg kasnijeg rada i pronalaska
- U memoriju WAL-a, uḿožemo učitati samo poslednji segment
- Poslednji segment možemo i dodatno markirati — dodati marker *_END*

Format koji mi koristitimo sličan je RocksDB-u, ali malo uprošćen, znači mi možemo da definišemo veličinu segmenta kako nama odgovara

```
+---------------+---------------+---------------+---------------+-----------------+...+...+
|    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
```

Kako segmentirati naš uprošćen WAL format, ideje :) ?

Segmetacija - mogućnosti

- Prag možemo definisati na nekoliko načina
- Po veličini
- Po broju zapisa
- Po transkacijama
- ...

Uporediti

- Ako segmente podelimo na istu veličinu, onda smo mi dužni da višak podataka prebacimo u naredni segmet
- Dužni smo da mi vodimo računa o ovome, i da to ispravno implementiramo u algoritam.
- Prednost, sistemi koji koriste segmente iste veličine mogu da se lakše optimizuju
- Ako definišemo da je segment racličite veličine, lakše je za implementaciju, ali optimizacija je teža
- Ali možemo da imamo datoteku za recimo svaku transkaciju
- To nas dovodi do toga da imamo dosta malih segmenata na disku — zauzimaju prostor

Brisanje delova loga

- Prethodnom idejom rešili smo problem performasni našeg WAL-a
- Velika datoteka neće više biti uskor grlo sistema
- Ali sada možemo dobiti jako puno malih datoteka na disku
- Ako to ne rešimo, imaćemo problem da će male datoteke zauzeti sav disk!
- Rešimo jedan problem, pojavi se drugi :(

Write Ahead Log - Problem 1

Implementirali ste WAL mehanizam, i rešili ste problem uskog grla, ali sada ima dosta datoteka. Kako da elimišemo datoteke koje više nisu od važnosti za wal

Kako ovo rešiti, ideje :) ?

Low-Water Mark

- Treba nam mehanizam koji će reći WAL mašineriji koji deo segmenata je bezbedno obrisati
- WAL zna putanju do *wal* direktorijuma, odakle da briše segmente — ova odluka se pokazala korisna :)
- *Low-Water Mark* ideja daje najniži indeks ili **low water mark**, pre koga se segmeti mogu obrisati
- Kratko rečeno, to je indeks u WAL-u unapred definisan, koji pokazuje koji deo WAL-a može da se obriše
- Uglavom možemo obrisati sve osim poslednjeg segmenta

- Ako ovaj mehanizam radi u isto vreme kada i WAL, može doći do problema
- Zaustavićemo WAL mašineriju da bi obrisali segmetne (Java GC)
- Ovaj mehanizam se uglavom pokreće kao nezavisan proces.program (ili nit više o tome naredni semestar :)))
- Dakle, potreban je mehanizam koji neće zaustaviti rad WAL-a, ali već koji će u pozadini obrisati nepotrebne segmente
- U velikim sistemima baza podatka, ovo može da se radi kada se dešava *snapshot sistema*, zarad dodatne sigurnosti og greške i otkaza

Izmene sa više klijenata

- Do sada smo WAL posmatrali samo kao sistem koji ima jednog klijenta
- U sistemima baza podataka to je (uglavom) slučaj — baza je klijent za WAL
- Ali WAL se može koristiti i za druge aplikacije
- Sistemi zasnovani na WAL-u i log-u su jako popularni u zadnje vreme
- Tada možemo imati više korisnika koji nešto pišu/čitaju

- Ako ne vodimo računa, podaci mogu biti nekonzistentni
- Jedan klijent može da obriše ili promeni informacije drugog
- WAL može da se sačuva informacije u pravom redosledu, ali ne mora
- ...

Write Ahead Log - Problem 3

Kako da dodajemo ili menjamo informacije u WAL-u, a da inforamcije i dalje budu konzistentne?

Kako ovo rešiti, ideje :) ?

Serijalizacija posla

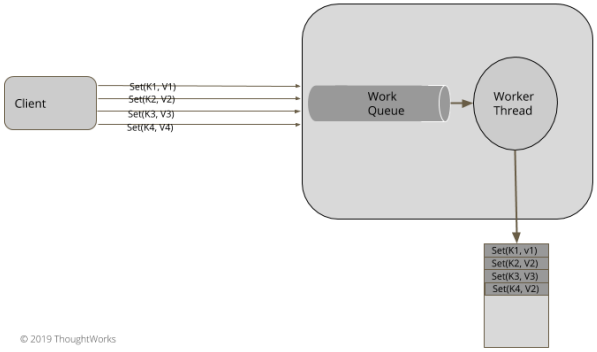
- Rešenje je relativno jednostavno
- Možemo napraviti red čekanja, i u njega dodavati poslove
- WAL će povremeno kupiti inforamcije iz ovog reda i upisivati ispravno kako su se oni dešavali
- Ovaj proces se još naziva i *serijalizacija* posla ili red čekanja
- Prednost ove ideje je što klijent brže dobija odgovor nazad, i može da nastavi sa poslom
- **ALI** uvek treba razmisliti o trajnosti podataka, da li i taj red treba da se perzistira na disk

Write Ahead Log - Problem 4

Uvek treba razmisliti o trajnosti podataka, da li i taj red treba da se perzistira na disk?

Ideje :) ?

Kratak odgovor bi bio, zavisi od tipa aplikacije i okruženja u kom se izvršava :)



(Martin Fowler Singular Update Queue

https://martinfowler.com/articles/patterns-of-distributed-systems/singular-update-queue.html)

Dodatni materijali

- Write-Ahead Log for Dummies (nije uvreda :))
- Write Ahead Log Martin Fowler
- Database Internals: A Deep Dive into How Distributed Data Systems Work
- Read, write and space amplification
- ARIES/NT: A Recovery Method Based on Write-Ahead Logging for Nested Transactions
- Linux mmap OS call
- Exploring mmap using go

Write Ahead Log - Pitanja

Pitanja :) ?