

Napredni algoritmi i strukture podataka

Otkaz i oporavak, Sekvencijalno i nasumično čitanje i pisanje

Strukture zasnovane na log-u, Write Ahead Log



Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Otkaz i oporavak

- ▶ Mreža je vrlo nepouzdana
- ▶ Čvorovi otkazuju nezavisno jedan od drugog
- ▶ Njihovi diskovi ne moraju nužno da otkazu
- ▶ To možemo da iskoristimo da rekonstruišemo stanje sistema, u slučaju otkaza
- ▶ Cilj nam je da postignemo mogućnost povratka stanja sistema, nakon otkaza, tj. nakon ponovnog pokretanja sistema

Kako to da postignemo, ideje :) ?

Sekvencijalno i nasumično čitanje i pisanje

- ▶ Ulazno izlazne operacije (I/O) na računaru, nisu stvorene jednake!
- ▶ Ovde se pod I/O operacijama misli na operacije koje se izvode nad diskom računara
- ▶ Da bismo razumeli razliku u performansama I/O operacija, zamislite kancelariju veterinaru
- ▶ Svi podaci se skladište na papiru u ormarićima za datoteke
- ▶ Za jednu životinju, sve informacije se čuvaju u različitim fasciklama
- ▶ Fascikle su smeštene u različite ormare prema nekim kategorijama

Pitanje

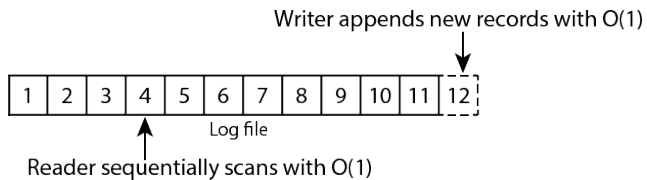
- ▶ Koliko je jednostavno pronaći potrebne informacije?
- ▶ Da li bi bilo lakše da se svi podaci nalaze u jednoj datoteci, tako da možemo da ih preuzmemo u jednom koraku?

- ▶ Ovo je u osnovi razlika dva pristupa: **(1)** nasumičnog (random) I/O-a i **(2)** sekvencijalnog I/O-a
- ▶ Nasumično pristupanje podacima je mnogo sporije i manje efikasno od sekvencijalnog (uzastopnog) pristupa
- ▶ Jednostavno gledano, brže je pisati/čitati iste podatke sa jednim uzastopnim I/O, nego više manjih nasumičnih I/O operacija
- ▶ Za ovo postoje jednostavne strukture podataka na koje se možemo osloniti

Strukture zasnovane na log-u

- ▶ Osnovna organizacija podataka je *log* (dnevnik) — niz *append-only* zapisa
- ▶ Ideja je nastala 1980ih kao *Log Structured File System*, danas se ova ideja dosta koristi dosta kod baza podataka i sistema koji rade sa velikim količinama podataka
- ▶ Zapisi u *log-u* su nepromenljivi, i dodaju se u strogo u sekvencijalnom redosledu **na kraj datoteke**
- ▶ *Log* je struktura podataka sa konstantnom $\mathcal{O}(1)$ brzinom pisanja/čitanja
- ▶ *Log* je efikasan na HDD i SSD diskovima
- ▶ Brzina se ne smanjuje čak i ako dnevnik ima terabajte podataka
- ▶ Uzmite to u obzir jer su u *cloud-u* HDD-ovi mnogo jeftiniji od SSD-ova

- ▶ Čitanje $\mathcal{O}(1)$
- ▶ Pisanje $\mathcal{O}(1)$
- ▶ Pretraga $\mathcal{O}(n)$ — full scan



(Log data structure, <https://scaling.dev/storage/log>)

Dodatni materijali

- ▶ Log data structure
- ▶ Log Structured File System for Dummies (nije uvredljivo :))
- ▶ Log Data Structure USENIX

Strukture zasnovane na log-u - Pitanja

Pitanja :) ?

Problem

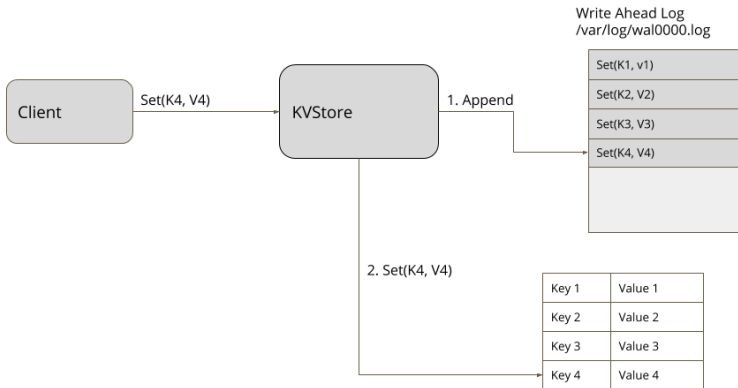
Od vas se zahteva da implementirate sistem za skladištenje podatka (u npr. Facebook-u), i pred vama su sledeći zahtevi:

- ▶ Snažna garancija trajnosti podataka je potrebna, čak i u slučaju da mašine otkažu
- ▶ Kada mašina pristane da izvrši operaciju, trebalo bi da to uradi čak i ako ne uspe ili se restartuje gubeći prethodno stanje u memoriji

Predlozi :) ?

Write Ahead Log - Ideja

- ▶ Kada se zapis upiše u neko skladište podataka, on se čuva na dva mesta: **(1)** Memorijska struktura (više o tome kasnije) i **(2)** Write Ahead Log (WAL).
- ▶ WAL deluje kao rezervna kopija na disku, za memorijsku strukturu tako što vodi evidenciju o svim operacijama nad skladištem podataka
- ▶ U slučaju ponovnog pokretanja sistema (restart), memoriska struktura se može u potpunosti oporaviti ponavljanjem operacija iz WAL-a
- ▶ Kada memorijska strukutra dostigne definisani kapacitet transformiše se u strukturu na disku (više o tome kasnije), WAL se briše sa diska da bi se napravio prostor za novi WAL



© 2019 ThoughtWorks

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

Write Ahead Log - Osobine

- ▶ Sekvencijalni I/O je brži od nasumičnih I/O operacija
- ▶ Sistem kao što su npr baze podataka treba da odgovare ovoj realnosti
- ▶ WAL koristi isključivo sekvencijalni I/O za skladištenje podataka na disku
- ▶ WAL kao struktura podatka, direktno se oslanja na strukturu zasnovanu na log-u
- ▶ Prosto gledano, WAL je jedan veliki log na disku

Write Ahead Log - Osobine

- ▶ Ovaj pristup ima i svoje nedostatke — *Nema besplatnog ručka!*
- ▶ WAL plaća svoju poboljšanu brzinu pisanja, dodatnim prostorom na disku
- ▶ Svaki put kada se zapis ažurira, stare verzije zapisa se čuvaju i zauzimaju dodatan prostor na disku
- ▶ Ovim postićemo da se podaci neće menjati, i u slučaju otkaza, možemo da reskonstruišemo zapise kako su se oni dešavali

Write Ahead Log - Space Amplification

- ▶ Odluka da se stare verzije čuvaju, u teoriji dizajna baze podataka je poznato kao *Space Amplification*
- ▶ Space Amplification je umnožak koliko prostora za skladištenje se koristi za datu veličinu skupa podataka
- ▶ Na primer, skup podataka od 1 GB sa faktorom od 2x bi rezultirao korišćenjem diska od 2 GB.
- ▶ Iako nije važno za vaš projekat, to je nešto o čemu su dizajneri baza podataka svesni i to šta se optimizuje

Write Ahead Log - Buffered i Unbuffered I/O

- ▶ Ovo je jedna od velikih tema za debatu u dizajnu baze podataka — baferovan nasuprot nebaferovan I/O
- ▶ Danas aplikacije zahtevaju više od baza podataka, dok diskovi ne drže korak sa ovim zahtevima
- ▶ Da bi se diskovi učinili bržim, OS mapira delove diska u memoriju (tema za sledeći put)

- ▶ Promene na disku se dešavaju samo u memoriji, a periodično OS upisuje promene na fizički disk
- ▶ Ovo je poznato kao baferovani I/O — zapisujemo podatke u bafer koji se na kraju isprazni na disk
- ▶ Baferovani I/O se može izbesci korišćenjem I/O bez baferovanja — podatke upisujemo direktno na fizički disk odmah kako dolaze
- ▶ Ovo može rezultovati prevelikim brojem operacija ka disku

Write Ahead Log - Struktura

Struktura može da se razlikuje od sistema do sistema, ali neka okvirna struktura bi sadržala sledeće elemente:

- ▶ Identifikator zapisa — id
- ▶ Niz bajtova koji reprezentuje podatke koji se zapisuju
- ▶ Informaciju da li je podatak dodat, ili obrisani (*Tombstone*)
- ▶ Vremensku odrednicu kada je zapis načinjen

Struktura može da se razlikuje primer je RockDB, popularan engine za NoSQL baze podataka:

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

(RocksDB Write-Ahead-Log-File-Format

<https://github.com/facebook/rocksdb/wiki/Write-Ahead-Log-File-Format>)

Format koji ćemo mi koristiti biće sličan RocksDB-u, ali malo uprošćen

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

*Podaci se čuvaju u binarnom obliku. Za čitanje, potrebno je ispravno prolaziti kroz binarni fajl i čitati podatke sa njihovih pozicija

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

Write Ahead Log - Pitanja

Pitanja :) ?