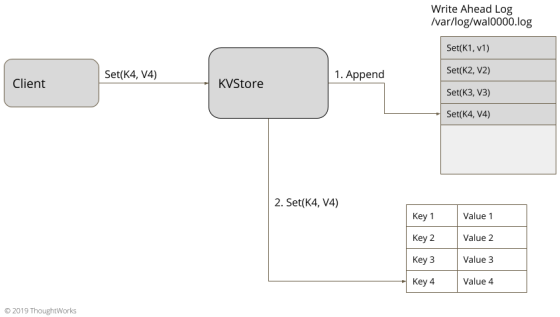
Napredni algoritmi i strukture podataka

Write-Ahead Log

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Write-Ahead Log

- WAL predstavlja rezervnu kopiju na disku za neku drugu strukturu, koja se nalazi u memoriji
- Vodi evidenciju o svim operacijama koje su se desile
- U slučaju ponovnog pokretanja sistema, memorijska struktura se može u potpunosti oporaviti/rekonstruisati ponavljanjem operacija iz WAL-a



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

- WAL je *append-only* struktura, prati vremenski tok rada sa podacima
- Noviji zapisi su na kraju WAL-a
- Stariji zapisi su na početku WAL-a
- Podatke u WAL možemo da dodamo samo dodavanjem na kraj strukture
- Izmena ili brisanje nekog podatka rezultuje novim zapisom u WAL, *in-place* izmena nije moguća
- Kada čitamo podatke, možemo da čitamo od početka ili da skeniramo od nekog dela

Format jednog zapisa

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

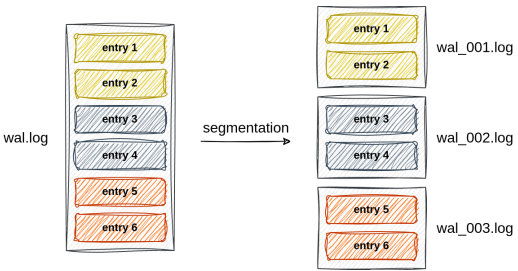
```
+---------------+-----------------+---------------+---------------+-----------------+-...-+--...--+
|    CRC (4B)   | Timestamp (16B) | Tombstone(1B) | Key Size (8B) | Value Size (8B) | Key | Value |
+---------------+-----------------+---------------+---------------+-----------------+-...-+--...--+
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
```

CRC

- CRC koristimo kao **error-detecting** mehanizam za otkrivanje promena u podacima
- CRC je *hash* funkcija koja detektuje promene nad podacima
- Kao tip kontrolnog zbira (checksum), CRC proizvodi skup podataka fiksne dužine na osnovu izvnorne datoteke ili većeg skupa podataka
- Ovo možemo koristiti kao mehanizam potvrde da li je bilo izmena/oštećenja kada se podaci pročitaju
- Ako je došlo do promene, taj podatak više nije validan

Segmentirani WAL

- Kako bismo lakše uklanjali stare zapise koji nam više nisu potrebni, možemo nekako da podelimo WAL datoteku - **segmenti**
- Za svaki segment možemo da specificiramo veličinu
- Sadržaj segmenata ne držimo u memoriji, već ih čitamo oslanjajući se na seek operaciju



- Uvek treba da znamo redosled segmenata, kako bismo ispravno utvrdili redosled događaja/operacija
- Segmentacijom WAL-a moramo obezbediti jednostavan način za mapiranje offset-a WAL-a (ili rednih brojeva) u segmente
- Na primer, ime svakog segmenta se dobija spajanjem unapred poznatog prefiksa (npr wal) i offset-a (ili rednog broja segmenta) — npr: wal_0001.log
- Kada se sistem pokrene, treba da preskenira *wal* direktrijum i pokupi lokacije (putanje do) segmenata

Brisanje delova WAL-a

- Treba nam mehanizam za brisanje segmenata WAL-a koji nam više nisu potrebni
- WAL zna putanju do *wal* direktorijuma, odakle će da briše segmente
- *Low-Water Mark* ideja daje najniži indeks ili **low water mark**, pre koga se segmeti mogu obrisati
- Kratko rečeno, to je indeks koji zadamo WAL-u, koji pokazuje koji deo WAL-a može da se obriše

UNIX sistemski poziv mmap

- mmap je veoma koristan alat za rad sa I/O
- mmap je sistemski poziv, što znači da brigu oko sinhronizaijce i swap-a prepuštamo onome ko to radi dobro i efikasno — OS
- Iz perspektive programera, čitanje pomoću mmap-irane datoteke izgleda kao normalna operacija pokazivača i ne uključuje dodatne pozive
- Dosta se koristi u dizajnu baza podataka

Zadaci

- Implementirati WAL strukturu i kao format zapisa koristiti format naveden u helper fajlu
- Omoguíti sledeće operacije
  - Dodavanje novog zapisa u WAL datoteku
  - Čitanje svih zapisa iz WAL-a
  - Čitanje jednog po jednog zapisa (morate čuvati informaciju o tome šta je poslednje pročitano)
- Omogućiti sledeće načine zapisa u WAL datoteku:
  - Svaki zapis treba da se zapiše direktno u fajl
  - *batch* zapis, tako što ćete napraviti nekakvu *buffer* strukturu u memoriji (koristeći strukturu po izboru), kada se napuni kapacitet (vi definišete) dodati zapise u WAL datoteku

Zadaci

- WAL treba da bude podeljen na segmente, svaki segment treba da sadrži n zapisa, gde n birate sami
- Napraviti folder *wal* i u njemu čuvati segmente WAL-a
- Segmente imenovati u formatu *prefx_offset.log* (npr. wal_00001.log, wal_00002.log)
- WAL prilikom pokretanja treba da preskenira *wal* folder i da zabeleži sve segmente i njihove lokacije
- WAL treba da sadrži low water mark index, tj. do kog indeksa je bezbedno brisati segmente
- Napraviti funkciju koja koristi low water mark i iz *wal* foldera briše dopuštene segmente. Nakon brisanja treba preimenovati WAL datoteke tako da offset-i budu ispravni