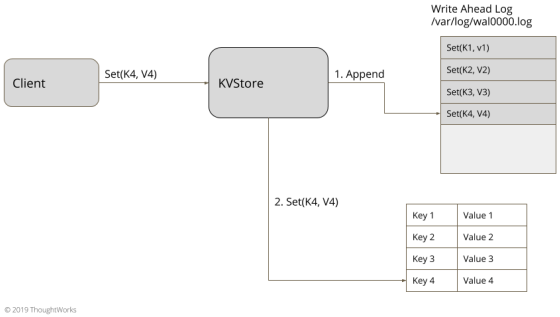
Napredni algoritmi i strukture podataka

Write-Ahead Log, Block manager, Buffer pool

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

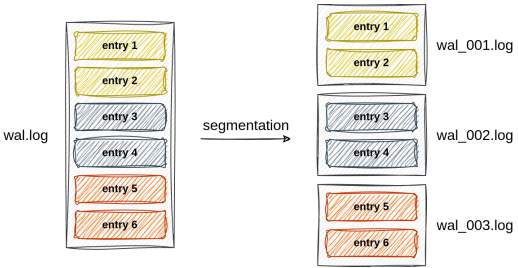
```
+---------------+---------------+---------------+---------------+-----------------+-...-+--...--+
|    CRC (4B)   | Timestamp (16B) | Tombstone(1B) | Key Size (8B) | Value Size (8B) | Key | Value |
+---------------+---------------+---------------+---------------+-----------------+-...-+--...--+
CRC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



- 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
- Međutim, kako je sva kontrola prepuštena OS-u, ne možemo uvek optimizovati operacije spram potreba sistema

Blokovska organizacija diska

- Disk je segmentiran na stranice fiksne veličine (512B-16KB).
- Najmanji deo diska nad kojim se obavljaju IO operacije je jedna stranica.
- Blok je logički segment čija veličina je umnožak veličine stranice.
- Oslanjajući se na ovu činjenicu, sve operacije čitanja ili pisanja na disk treba da se obavljaju nad blokovima, a ne nad pojedinačnim zapisima.

Block manager

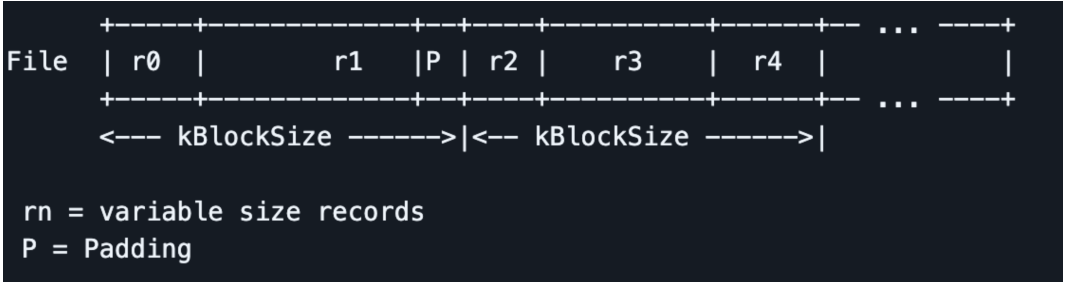
- Struktura koja podržava operacije čitanja i zapisa bloka.
- Sve ostale strukture kojima je potreban disk (za sada WAL), pristupaju mu preko block manager-a, **ne direktno.**
- Block manager oslanja se na **buffer pool** (block cache) strukturu.
- Buffer pool u memoriji čuva blokove kojima je sistem skoro pristupao.

Block manager operacije

- Kada neko pozove operaciju **čitanja** nad block manager-om, on prvo proveri da li je blok dostupan u buffer pool-u i, ako jeste, vrati taj blok.
- Ako se blok ne nalazi u buffer pool-u, čita se sa diska, upisuje u buffer pool i vraća onome ko je pozvao operaciju.
- Pri pozivu operacije **pisanja**, blok se zapisuje u buffer pool.
- Pošto nemamo neograničeno memorije i želimo da se podaci u nekom trenutku perzistiraju, blokove iz buffer pool-a prebacujemo na disk kada on prekorači definisanu veličinu.
- Kasnije u toku semestru pričaćemo o algoritmima za izbacivanje blokova iz pool-a, za sada je ovaj pristup dovoljan.

Padding bloka

- Jedan blok se sastoji iz jednog ili više WAL zapisa
- Kada dodajemo novi WAL zapis u blok, ako on ne može u celosti da stane u taj blok, zapisujemo ga u naredni.
- Preostali prazan prostor u bloku treba popuniti praznim (null) vrednostima - **padding**



Fragmentacija korisničkog zapisa

- Jedan korisnički zapis može da se prostire kroz jedan blok, ali može i kroz više njih
- U tom slučaju, jedan korisnički zapis delimo na više WAL zapisa uvodeći polje **type** u zapis, koje može imati sledeće vrednosti: **FULL, FIRST, MIDDLE, LAST**
- **FULL** - Čitav korisnički zapis nalazi se u jednom WAL zapisu
- **FIRST, MIDDLE, LAST** - Korisnički zapis podeljen je u više WAL zapisa, fragmenata, zbog granica bloka

```
+---------+-----------+-----------+---------------+--- ... ---+
|CRC (4B) | Size (2B) | Type (1B) | Log number (4B)| Payload   |
+---------+-----------+-----------+---------------+--- ... ---+
Same as above, with the addition of
Log number = 32bit log file number, so that we can distinguish between
records written by the most recent log writer vs a previous one.
```

Zadaci

- Implementirati WAL strukturu čiji zapisi imaju format definisan u helper fajlu (opciono, zapis može imati i size atribut nakon CRC atributa).
- Omoguíti sledeće operacije
  - Dodavanje novog zapisa u WAL datoteku
  - Čitanje svih zapisa iz WAL-a
- Implementirati block manager i buffer pool strukture. Za sav posao na disku, WAL se oslanja na block manager-a. Proširiti WAL zapise **type** atributom i podržati padding i fragmentaciju.
- WAL treba da bude podeljen na segmente, svaki segment treba da sadrži n blokova, gde n birate sami. Opciono, WAL zapisi mogu se proširiti poljem **Log number**.
- Podržati operaciju brisanja segmenata kojoj se prosleđuje low watermark indeks, do kog će segmenti biti obrisani