

# Napredni algoritmi i strukture podataka

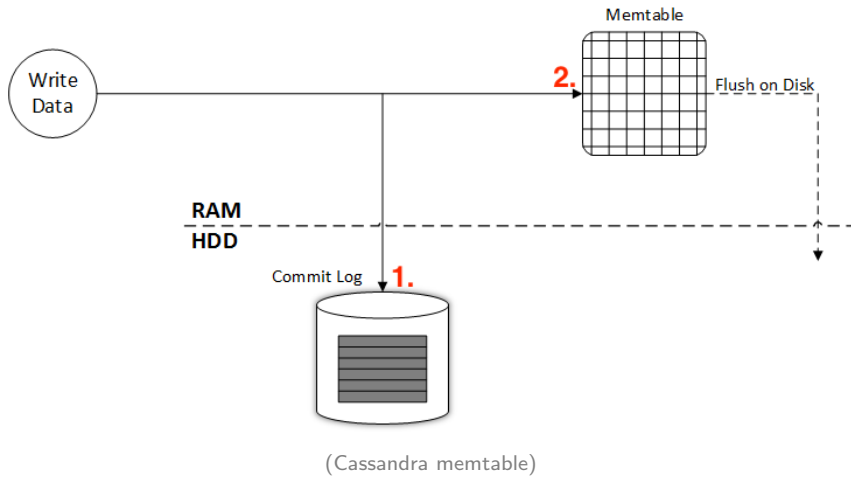
Memorijske tabele (Memtable), Eksternalizacija konfiguracije, Write path



**Univerzitet u Novom Sadu**  
**Fakultet Tehničkih Nauka**

## Memorijska tabela - ideja

- ▶ Ideja iz Memorijske tabele (Memtable) je relativno jednostavna — zapisati podatke u memoriju i čitati podatke iz memorije
- ▶ **AKO** se podaci nalaze u memoriji, sve operacije su relativno brže nego da su podaci **striktno** na disku
- ▶ Memorija je brza, memorija je super, memorija je kul, svi vole memoriju
- ▶ Memorija je aktivna dok je sistem aktivan
- ▶ **ALI** memorija nije sigurna :/
- ▶ Zato sistem komunicira sa WAL-om prvo, koji nam daje ove garancije, pa onda zapisuje u Memtable



## Memorijska tabela — struktura podataka

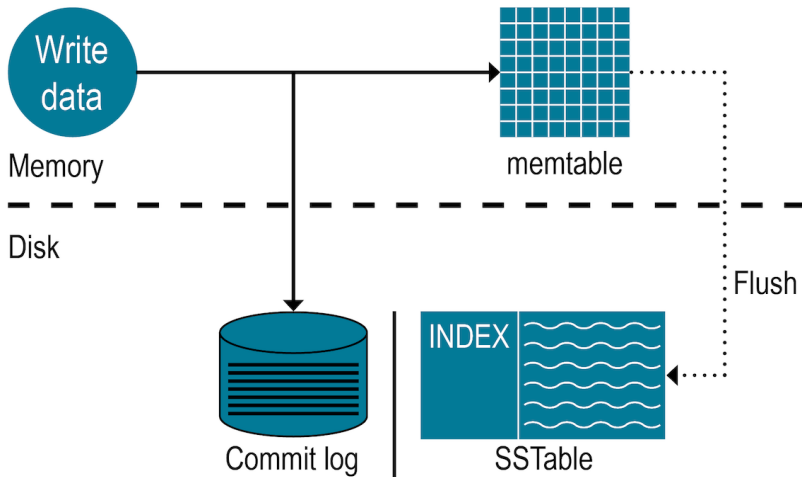
- ▶ Jednostavna struktura koju smo radili, i koja se dosta koristi za Memtable je *SkipList*
- ▶ RocksDB i LevelDB na primer direkno koriste SkipList
- ▶ Pored nje, u projektu ćemo kao alternativnu strukturu za Memtable ponuditi i B stablo (korisnik specificira kroz konfiguraciju)

## Memorijska tabela — zapis na disk

- ▶ Memtable se implementira kao struktura **fiksne kapaciteta**
- ▶ Setimo se segmenata i WAL-a, jedan segment može biti veličine kao i Memtable
- ▶ To ne znači da će kraj segmenta uvek značiti i potpuno popunjenu memorijsku tabelu, možda smo više puta menjali ili brisali vrednost pod istim ključem
- ▶ Kada se Memtable struktura popuni, prekorači se granica, ona se perzistira na disk — operacija **Flush**

## Putanja zapisa — Algoritam

1. Korisnik je poslao zahtev — nekakvu operaciju (dodavanje, čitanje, izmena, brisanje — CRUD)
2. Podatak se prvo zapisuje u **WAL**
3. Kada WAL potvrdi zapis, podatak se zapisuje u **Memtable**
4. Koraci (2) i (3) se ponavljaju dokle god ima mesta u Memtable-u
5. Ako je kapacitet Memtable-a popunjen, Memtable sortira parove ključ-vrednost
6. Sortirane vrednosti se zapisuju na disk formirajući **SSTable**
7. Možemo isprazniti **Memtable** ili napraviti nov, a prethodni uništiti ili rotirati



(Cassandra write path)

# Eksternalizacija konfiguracije

- ▶ Kada pravimo sistme koji se konfigurišu kroz eksterne fajlove, trebamo obezbediti podrazumevane vrednosti — **default**
- ▶ Ovo možemo da uradimo na dva mesta, da se osiguramo i zaštitimo od potencijalnih problema
  1. Obezbediti fajl sa default vrednostima — isti fajl za konfiguraciju samo već popunjen vrednostima
  2. **AKO** takav fajl ne postoji, obezbediti da kroz kod postoje default opcije koje program može da iskoristi
- ▶ Na ovaj način postićemo redundantnost i sistem nam je stabilniji
- ▶ Ovo nije obaveza, ali je generlano lepa praksa



## Zadaci

- ▶ Implementirati Memtable gde se kao struktura podataka koristi SkipList ili B stablo
- ▶ Napraviti saradnju sa Write Ahead Log-om, tako da se podaci **prvo** zapišu tu, a kada se dobije potvrda zapisa, podaci se **onda** zapišu u Memtable
- ▶ Elementi koje Memtable čuva treba da sadrže ključ, vrednost, timestamp i tombstone
- ▶ Omogućiti da se veličina Memtable-a, WAL segmenta i struktura memorijske tabele specifiiraju kroz YAML konfiguracioni fajl
- ▶ Kada se Memtable popuni, sortirati vrednosti po ključu i ispisati na ekran, a zatim obrisati podatke iz tabele
- ▶ Memtable prihvata dodavanje; izmenu **AKO** je podatak sa tim ključem prisutan, ako nije uraditi dodavanje; brisanje je logičko, postavite vrednost **tombstone** parametra na true