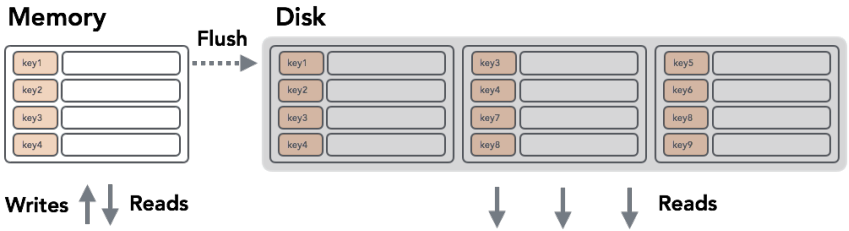
Napredni algoritmi i strukture podataka

Put čiranja podataka (Read path)

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Put čitanja podataka — Read Path

Write path smo videli na nekom od prethonih predavanja, Read path ukratko prošlo predavanje, ali jako prosto



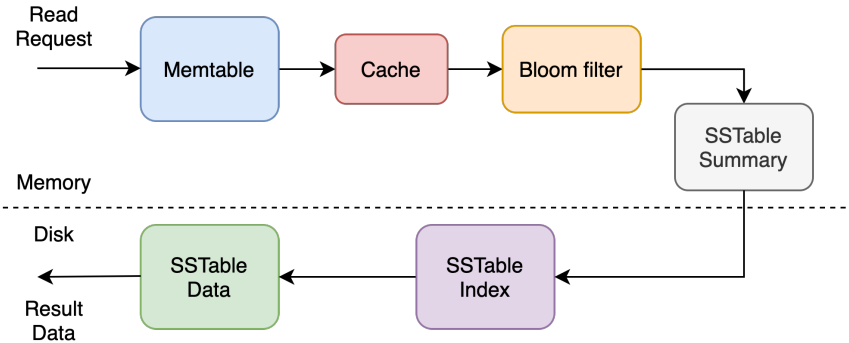
(Databases Internals)

- Moramo učitati **Bloom filter** sa diska i videti da li je ključ **možda** tu
- Ako nije, odmah javimo korisniku, da ključ nije prisutan — podatak nije sačuvan
- Ako je **možda** tu, učitati **summry** i videti da li je ključ u tom opsegu
- Ako nije, odmah javimo korisniku, da ključ nije prisutan — podatak nije sačuvan
- Ako **jeste**, pronaći ga u **index** strukturi, i uzeto **offset**
- Kada imamo **offset**, možemo da se pozicioniramo na **Data** deo i da pročitamo podatak i vratimo korisniku

- Svi pretohodno formirani elementi su nam potrebni, da bi što pre stigli do podatka koji tražimo, **AKO** je on tu
- Toliki broj fajlova nam treba zbog ovog **AKO**
- Pravimo kompenzaciju za nesigurnost Bloom Filter-a
- Što je pre moguće da dobijemo informaciju nazada
- Ili bar da nam mašinerija kaže: **ključ koji tražis nije sigurno tu**
- Svi formirani elemnti čine put čitanja podataka – **Read Path**

- I svi oni moraju da se povežu u sinhronu celinu, **AKO** želimo da dobijemo informaciju u razumenom vremenu
- Podatke u cache-u moramo da ažurirmao svaki put kada se korisnički podatak locira!
- Kada dobijemo podataka, pre nego što ga vratimo korisniku prvo ga zapišemo u cache
- Ova prosta strategija nam omogućava da kod sledeće pretrage **MOŽDA** ne moramo da idemo po disku
- Kada se uputi zahtev za brisanje nekog podatka, **AKO** je on u cache-u, možemo ga obrisati

Read path


Read
Request
Memtable
Cache
Bloom filter
SSTable
Summary
Memory
Disk
SSTable
Data
SSTable
Index
Result
Data

Zadaci

- Povezati sve prethodno kreirane elemente u jednu sinhronu celinu
- Implementirati **Read Path** strategiju
- Dodati GET operaciju koja će inicirati **Read Path** strategiju