SKALABILNOST

Tim 10 - Jovan Jelicki, David Drvar, Tamara Kovačević, Ilija Brdar

1. Dizajn šeme baze podataka (konceptualni, logički ili fizički)

Dizajn šeme baze podataka dat je kroz UML Class dijagram koji je okačen na git repozitorijumu na grani class-diagram.

2. Predlog strategije za particionisanje podataka

Kako je ukupan broj korisnika aplikacije blizu 200 miliona i broj zakazanih pregleda na mesečnom nivou oko milion, to je previše podataka koji bi se čuvali na jednoj mašini. Zbog toga je potrebno distribuirati podatke na više mašina, kako bi se operacije nad njima obavljale brže. Strategija za to može biti particionisanje na osnovu ID-a pregleda i vremena kad je zakazan pregled. U tom slučaju, ID pregleda će se sastojati od predefinisanog ID-a i vremena kad je zakazan. Na taj način, možemo unapred da znamo na kojim serverima će se čuvati pregledi koji su najbliži sadašnjem trenutku (uskoro će se korisniti) i brzo ih dobaviti. Pored servera sa najbližim pregledima, možemo da definišemo hash funkciju (sa ID argumentom) koja će nam u konstantnom vremenu reći na kom serveru se nalazi traženi pregled. Ovim se rešava problem pretrage milion pregleda (samo na mesečnom nivou) koji bi se dešavao u O(n) vremenu.

3. Predlog strategije za replikaciju baze i obezbeđivanje otpornosti na greške

Možemo upotrebiti multimaster arhitekturu sa više “glavnih” baza podataka zbog velikog broja funkcionalnosti koje zahtevaju upisivanje u bazu (rezervacije lekova, pregleda, ocenjivanje, ponude, žalbe...). Koristila bi se *Partial Replication* šema replikacije podataka, što znači da se ne bi na svim bazama nalazili isti podaci (jer za tim nema potrebe), već oni koji su neophodni u određenom prostoru i vremenu. Ukoliko bi došlo do greške, uloga primarnog servera bi se prebacila na neki od sekundarnih.

4. Predlog strategije za keširanje podataka

Sa obzirom na količinu podataka koju prati tako velik broj korisnika, neophodno je omogućiti brzo dobavljanje podataka kroz keš mehanizam. Prvobitno, potrebno je kategorisati podatke po načinu njihovog korišćenja. Za one podatke koji se često samo čitaju bez promene - prošli izveštaji, završeni pregledi, otkazani pregledi, listu alergija, neželjenih efekata - koristili bismo READ_ONLY strategiju. Ostali podaci koji su podložni čestim upisima kao što su slobodni pregledi, rezervacije lekova, količine lekova u apoteci bili bi keširani READ_WRITE konkurentnom strategijom. Kako je za više nivoe keširanja neophodan eksterni provajder, koristili bismo EhCache koji podržava mnoge strategije keširanja što ga čini dobrim izborom ukoliko ga je potrebno prilogođavati sistemu u budućnosti. Još jedna od pogodnosti ovog provajdera je mogucnost definisanja koliko dugo će objekat postojati u keš memoriji, što bismo uradili za neke podatke kao sto su završeni i otkazani pregledi koji se retko dobavljaju.

5. Okvirna procena za hardverske resurse potrebne za skladištenje svih podataka u narednih 5 godina

Uzevši u obzir kompleksne entitete našeg sistema, odredićemo petogodišnju procenu potrebe za hardverskim resursima.Kako se u našoj aplikaciji na mesečnom nivou održi 1 000 000 pregleda, a ako pretpostavimo da je za čuvanje jednog pregleda potrebno 160B, možemo doći do zaključka da se na godišnjem nivou održi 1 000 000*110B*12 pregleda.Ako uzmemo u obzir da aplikaciju koristi 200 000 000 korisnika,a da je za čuvanje kredencijala korisnika potrebno 120B dolazimo do podatka da je za skladištenje naših korisnika neophodno izdvojiti 200 000 000*80B.Svaki korisnik ima mogućnost pisanja žalbi, pretpostavimo da na mesečnom nivou pristigne 1000 žalbi, gde je maksimalna dužina svake od njih 100 karaktera.(veličina UTF-8 zauzima 4B).U okviru naše aplikacije vodimo računa i o količini lekova na stanju, ako pretpostavimo da u svakom trenutku posedujemo 6000 različitih vrsta lekova, a da je za čuvanje jednog leka neophodno 90B, lekovi nam zauzimaju 6000*90B. Mesečni broj rezervacija lekova je 1 000 000, ukoliko pretpostavimo da jedna rezervacija zauzima 45B, na rezervacije biva utroseno 1000000*45*12B na godišnjem nivou.Uzevši u obzir preostalih 40 entiteta i njihovu veličinu, koja u proseku iznosi 50mb,zaključujemo da će nam za iste biti potrebno 50*40mb. Uzevši u obzir ove podatke, možemo doći do zaključka da nam je za petogodišnje čuvanje podataka potrebno 5*(1000000*160*12+1000*12*100*4+1000000*45*12)+200000000*120+6000*90+1.95=28.24GB

6. Predlog strategije za postavljanje load balansera

Load balanser možemo postaviti između klijenta i pool-a aplikativnih servera. Radi opsluživanja što većeg broja korisnika u što kraćem vremenu, možemo primeniti Least time tehniku balansiranja. Ova tehnika podrazumeva postojanje funkcije koja će kao svoj rezultat dati broj servera kom se šalje zahtev, na osnovu trenutnog broja konekcija sa serverom i brzine odgovora servera (bira onaj sa najmanjib brojem konekcija i najbržim odgovorom).

7. Predlog koje operacije korisnika treba nadgledati u cilju poboljšanja sistema

Broj novih pregleda po danu, broj žalbi po danu, period dana kada je najveci saobracaj, najbolje ocenjeni lekari, lekovi, apoteke (za bolji recommender sistem), broj rezervacija nekog određenog leka u određenom vremenskom periodu

8. Kompletan crtež dizajna predložene arhitekture (aplikativni serveri, serveri baza, serveri za keširanje, itd)