Student 2 – Danica Vojvodić RA 27/2017
Student 3 – Jelena Vlajkov RA 32/2017

Skalabilnost

Pretpostavke:

- ukupan broj korisnika aplikacije je 200 miliona,

- broj rezervacija lekova i zakazanih pregleda kod farmaceuta i dermatologa na mesečnom nivou je milion,

- sistem mora biti skalabilan i visoko dostupan.

1. Dizajn šeme baze podataka

Dizajn šeme baze podataka je okačen u ovom direktorijumu u pdf format zbog preglednosti ovog dokumenta.

2. Predlog strategije za particionisanje podataka

Da bi se postigle dobre performanse i visoka dostupnost podataka pri većim opterećenjima servera, jedna od čestih praksi jeste particionisanje podataka.

Tabele koje potencijalno mogu imati veliki broj entiteta možemo horizontalnim particionisanjem podeliti u više manjih, grupisanjem po nekom ključu. U slučaju naše aplikacije to mogu biti tabele workdays koje možemo podeliti na osnovu apoteke, appointments koje možemo podeliti na osnovu tipa (Examination ili Counseling), pharmacyStorage koje takođe možemo podeliti na osnovu apoteke. Takođe, workdays i appointments možemo profiltrirati na osnovu datuma pod pretpostavkom da će se za workdays i appointments koji su prošli, odnosno čiji je datum stariji od trenutnog mnogo ređe vršiti upiti, te kvalitetniji hardware dodeliti za one čiji su datumi posle trenutnog i za koje će sigurno postojati mnogo više frekventnih upita i time poboljšati performanse.

Praksu vertikalnog particionisanja možemo primeniti ponovo kod appointments i workdays, particioniranjem izdvajanjem perioda, odnosno datuma početka i završetka, za koji imamo mnogo frekventnije upite, odnosno ta dva podatka nam se koriste u jako velikom broju provera te bismo time ubrzali dodavanje radnih dana i zakazivanje samih pregleda. Sličnu stvar možemo uraditi i kod pricelists (jedan entitet u cenovniku apoteke koji ima svoj period važenja) ili drugReservations.

Takođe, možemo izdvojiti osetljive podatke kao što su lozinka korisnika u posebnu tabelu nad kojom kasnije možemo dodati dodatnu zaštitu i time poboljšati sigurnost samih podataka u našoj aplikaciji, a takođe bi se i upiti smanjili jer osetljive informacije se ređe koriste, dok se osnovne informacije o korisnicima koriste mnogo češće.

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

Predlog je klasterovati usere po njihovom geografskom položaju i u skladu sa finansijskim resursima obezbediti par servera od kojih je svaki blizu određenoj grupi korisnika (npr. po jedan server za Aziju, Evropu, Ameriku) i time poboljšati performanse brzine pristupa. Serveri se povezuju mrežom i imamo repliciranu bazu u svakom.

Najiscrpniji i najzahtevniji za resurse u našem sistemu su svakako appointmenti i drugReservations (pa tako i pharmacyStorage), pošto će aplikacija u najfrekventnije biti upotrebljava upravo za to - online rezervisanje pregleda i lekova. Svaki od servera obrađuje update i insert upite i iste prosleđuje ostalima kako bi bili u konzistentnom stanju. Kod appointment-a i drugReservations-a upotrebićemo full table replication jer su u pitanju inserti, dok kod pharmacy storage možemo upotrebiti key-based incermental replication jer update-ujemo samo po jedno polje u tabeli, u ovom slučaju količinu leka u skladištu.

Postojanje većeg broja servera i repliciranih baza na istim osigurava nas da u slučaju otkaza ili katastrofe neke druge prirode ostanemo bez podataka.

4. Predlog strategije za keširanje podataka

Praksu keširanja podataka treba sprovesti nad podacima koji se ne update-uju često i koji pretstavljaju veći napor za dobaviti (veći objekat, veći broj atributa). U slučaju naše aplikacije možemo keširati podatke kao što su podaci o apoteci, podaci o lekovima, podaci o samim userima, promocijama i cenovnicima apoteke, jer za iste ne očekujemo česte update-ove, te nema potrebe da ih svaki put dobavljamo iznova iz baze. Čak i u slučaju update-a, možemo update-ovati cache. Keširanje će nam pomoći i za buduće eventualno skaliranje, jer će osloboditi server od nepotrebnih upita i time i smanjiti potrebu za skupim serverom snažnih performansi.

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

Uzevši u obzir početne pretpostavke:

- 200 000 000 korisnika - 0.47kb u proseku za skladištenje jednog korisnika.

200 000 000 * 0.47kb daje približno 100gb

- Od 200 000 000 korisnika 90% su sigurno pacijenti, a svaki pacijent imaće svoj medical-record koji u proseku za skladištenje zauzima 0.25kb

200 000 000 * 0.9 * 0.25kb daje približno 45gb

- 1000 - 2000 lekova u bazi - takođe oko 0.5kb za skladištenje jednog

2000 * 0.5kb daje približno 1gb

- Milion rezervacija i pregleda na mesečnom nivou – oko 0.15kb - 0.2kb po rezervaciji ili leku.

1 000 000 * 60 meseci *0.15kb daje približno 9gb

- Od 200 miliona korisnika,neka je 5% farmaceuta ili dermatologa – 10 000 000 medicinskog osoblja, od kojeg očekujemo u proseku 30 entiteta workday na mesečnom nivou (zbog višekratnog radnog vremena i rada u većem broju apoteka) - oko 0.15kb za skladištenje jednog entiteta workday.

10 000 000 * 30 * 60 meseci * 0.15kb daje priblizno 2.7TB za 5 godina

- Na 1 000 000 pregleda, neka je 10% otkazano. To ostavlja 900 000 entiteta izveštaja na mesečnom nivou, koji uzima prosečno 0.15kb

900 000 * 60 meseci * 0.15kb što daje približno 8.1gb

Ostali podaci uzimaju zanemarljivo malo u odnosu na prethodno navedene,pa cemo kao okvirnu procenu za njih dodati 10gb za 5 godina. Sve ovo zajedno kao okvirnu procenu hardverskih resursa potrebnih za skladištenje za 5 godina nam daje oko 5TB.

6. Predlog strategije za postavljanje load balansera

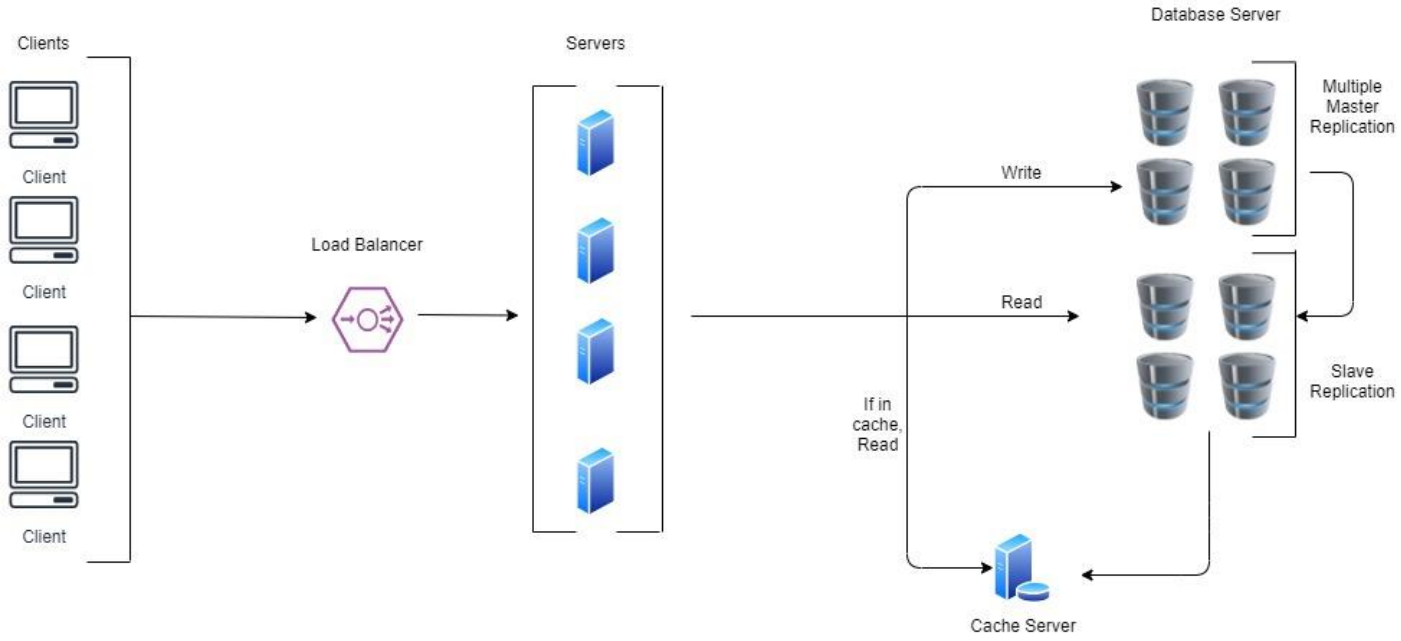
Kao predlog strategije za postavljanje load balansera, predložile bismo round robin algoritam. Takođe, ako bismo pretpostavile da server ne raspolažu ekvivalentnim konfiguracijama, tačnije da su u budućnosti planirani server različitih konfiguracija, tada bi se zahtevi raspoređivali ciklično uzimajući u obzir i težine čvorova (snažniji serveri bi prihvatali veći broj zahteva). Do tada bi se zahtevi raspoređivali ciklično po serverima bez ikakve prednosti određenih servera.

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

Kako bi sistem bio što bolji, uzimajući u obzir prirodu posla kojom se sistem bavi, sistem će biti više korišćen ako se ispune svi zahtevi pacijenata, tj. ako bi se dalji razvoj sistema bazirao isključivo na poboljšanju doživljaja pacijenata. To bismo postigli uvođenjem event sourcing-a. Kako pacijenti imaju pravo na pretragu i rezervaciju lekova, ako bismo beležili koje lekove su pacijenti najčešće pretraživali ili rezervisali, znali bismo koje lekove je najbitnije imati na stanju, te bi tada administratori znali koje lekove najčešće da naručuju. Ako bismo pratili koji farmaceuti i dermatolozi su najtraženiji na našem sistemu, znali bismo koje medicinsko osoblje bi trebalo najčešće plasirati sa pregledima.

Student 2 – Danica Vojvodić RA 27/2017
Student 3 – Jelena Vlajkov RA 32/2017

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



Kako smo već spomenuli da za smanjenje opterećenje servera je pogodno koristiti load balancer (round robin algoritam), što se tiče opterećenja poziva ka serverima baza najpogodnije bi bilo koristiti Multiple Master Replication i Slave Replication. Čitanje bi se isključivo vršilo iz Slave servera, osim ako se traženi podaci već ne nalaze u keš serveru, dok se upisivanje vrši preko Master servera, te se upis propagira na Slave servere.