Proof of concept

Dizajn šeme baze podataka

Dizajn šeme baze podataka nalazi se u fajlu d*bDesign.png*.

Particionisanje podataka, replikacija baze i obezbedjivanje otpornosti na greške

Sistem je distribuiran na više geografskih lokacija, gde se na svakoj lokaciji nalazi slave-master klaster baza podataka sa 3 slave baze , zajedno sa bekap bazom master baze. Nad globalnom bazom podataka bi odradili “sharding” koji bi bio zasnovan na geografskoj lokaciji i svaka master baza jednog geografskog klastera bi predstavljala jedan shard globalne baze.

Vertikalno skaliranje bi bilo odradjeno tako da bi se manje korišćene kolone tabela izdvojile u posebne tabele koje bi zatim bile čuvane na sporijim medijumima. Primer toga bi bilo izdvajanje *adress* i *phone_number* kolona iz tabele *staff* i *institution*, *gender* i *occupation* kolona iz tabele *blood_donor.*

Ovim pristupom bili bi ispunjeni Availability i Partitioning tolerance iz CAP teoreme dok bi bila postignuta eventualna konzistentnost, zbog toga što repliciranje podataka iz master u slave bazu zahteva odredjeno vreme.

Strategija za kesiranje podataka

Svaki klaster servera imaće svoj keš. Budući da naša aplikacija većinski radi sa veoma osetljivim ličnim podacima ili podacima koji se frekventno menjaju, skup objekata koje bi mogli keširati je veoma mali. Jedan od primera keširanja bi bile informacije o banakama krvi poput adrese, naziva, opisa... , zbog toga što se očekuje da će pacijenti često pretraživati njima dostupne banke krvi radi zakazivanja pregleda, a te informacije se retko menjaju. Za oslobadjanje koristila bi se strategija *Least Recently Used*, a pored nje preporučujemo posmatranje aplikacije neko vreme, te keširanje unapred najčešće posećivanih banaka krvi. Preporučujemo korišćenje Redisa.

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

Početni memorijski prostor koji nam je potreban otprillike je jednak 10mil * memorija potrebna da se memoriše jedan donor(5.3kB) + memorija potrebna da se uskladište sve ostale tabele, izuzimajući preglede. Kada se ovo zapažanje primeni na čitav sistem bazna memorija iznosi:

(53GB za donore + 500kB(memorija potrebna za ostatak baze)) *5(master, slave-ovi, bekap))(1 klaster) * broj geografskih klastera+ (53GB + 500kB) * broj geografskih klastera (globalna baza).

Pretpsotavićemo da imamo 10 geografskih klastera, te je bazna memorija 3.18TB.

Memorija potrebna da se zabeleži jedan pregled iznosi 15.6kB. Na mesečnom nivou očekuje se 500 000 rezervacija što je na nivou 5 godina jednako 30 000 000 rezervacija što zauzima 500GB (468GB za preglede + ostale tabele) . Kada se dovedu u obzir master-slave klaster i globalna baza ukupno zauzeće memorije iznosi: (500GB * 5) * 10 + 500GB * 10 = 30TB

Dakle procena memorijskog zauzeća u narednih 5 godina iznosi 33.18TB.

Load balanser

Zbog geografske raspodele klastera koristili bi jedan load balanser koji bi preusmeravao klijente na servere koji su im najbliži.

Serveri bi bili organizovani u active-active klaster. Time bi se distribuiralo opterećenje servera, kao i obezbediilo dalje funkcionisanje u slučaju pada jednog od servera. Odlučili smo se za active-active, pre nego za active-passive zbog toga što smatramo da je na taj način opterećenje bolje distribuirano.

Takodje, potrebno je postaviti load balanser izmedju klastera servera i klastera baza podataka. Ako je fokus na lakšoj implementaciji preporučuje se korišćenje *round robin* ili *random* tehnike raspodele pod uslovom da je hardver svih servera približno jednake moći, u suprotnom preporučuju se *weighted* verzije istih. Mi smo se na kraju ipak odlučili za *weighted least connection* tehniku, jer iako je teža za implementaciju njom se postiže mnogo balansiranija raspodela.

Budući da naš sistem koristi JWT, te je on *stateless*, nismo razmatrali tehnike replikacije sesije.

Predlog koje operacije korisnika treba nadgledati u cilju poboljšanja sistema

Predlažemo da se nadgledaju sve operacije korisnika vezane za preglede na dnevnom nivou, to jest zakazivanje, otkazivanje i vršenje istih zbog toga što one čine srž biznis logike aplikacije. Na osnovu tih informacija mogli bi, na primer, zaključiti na kojoj geografskoj lokaciji se povećao broj zakazivanja pregleda, te izvršiti vertiklano ili horizontalno skaliranje na tom mestu.

Dizajn predložene arhitekture

Dizajn predložene arhitekture nalazi se u fajlu *architetectureDesign.png*.