Skalabilnost

Skalabilnost je mogućnost aplikacije da ponese povećanje zahteva i broja korisnika a da sama aplikacija ne mora da se menja. Zahvaljući skalabilnosti postiže se uspešno rešavanje problema drastičnog pada performansi web aplikacija usled porasta broja posetilaca, kao i usporavnja uzrokovanog povećanjem broja servisa koji su u ponudi kao i kolicine podataka u opticaju. Što je aplikacija skalabilnija ona će lakše podneti povećan protok podataka. Cilj kojim teže svi projektanti sistema jeste da se postigne linearnost u brzini odgovora na zahtev i količine podataka sa kojima se manipuliše.

Skalabilan razvoj ne predstavlja posebnu vrstu razvoja sistema. To je set principa i tehnika koji daju smernice i uputstva za razvoja aplikacija višeg kvaliteta i standarda. Pridržavanjem tih pravila postiže se:

- redukovanje vremena učitavanja stranice, broj grešaka, vreme potrebno za implementaciju promena i trošak održavanja i unapređivanja aplikacije
- poboljšanje user experience-a
- produženje životnog ciklusa servisa i proizvoda
- poboljšanje prodaje i brenda

Imajući u vidu ove benefite u nastavku rada ćemo dati objašnjenja tehnika i pristupa koje bismo mi upotrebili da podignemo našu aplikaciju na viši nivo.

Kada web aplikacija dođe do stadijuma da povećan broj podataka sa kojima se manipuliše utiče na brzinu odgovora na zahtev, tj na učitavanje stranica, možemo reagovati na više načina:

- Uložiti novac u kupovinu hardvera koji će moći da se nosi sa novonastalom situacijom – vertikalna skalabilnost. Podrazumeva se sposobnost zamene komponenti na postojećem sistemu moćnijim i bržim, kako bi bi se ispratili povećani zahtevi.

- Misliti na vreme i dizajnirati samu aplikaciju tako da bude skalabilna, a to ćemo postići tako što ćemo na probleme odgvarati rešenjima koja podižu performanse. To se najviše odnosi na postizanje horizontalne skalabilnosti, koja podrazumeva podelu sistema na manje komponente i njhovo postavljanje na razlite masine, odnosno server koji izvršavaju funkcije simultano. Ovaj pristup zahteva izmene u programi kako bi se u potpunosti iskoristio povećan kapacitet resursa.

Tehike koje bismo upotrebili:

Load balancers

Moderni, komercijalni, globalni web sajtovi i aplikacije (kakav bi jednog dana postala i nasa 3Dots aplikacija) moraju da uslužuju stotine, hiljade pa i milione konkurentnih zahteva od klijenata ili korisnika i da vraćaju ispravan tekst, sliku, video ili podatke a

sve to na brz i pozdan način. Kako bi se to obezbedilo praksa je da se doda više servera, koji bi opsluživali ove zahteve, što bismo mi i primenili.

Load balanser se u u ovakvoj arhitekturi metaforčki rečeno ponaša kao policajac koji je pozicioniran ispred servera i usmerava klijentske zahteve ka onom koji bi bio najpogniji za njihovo izvršenje u pogledu maksimizovanja brzine i poboljšanja performanse.

U slučaju da se se jedan server ugasi, load balanser usmerava saobraćaj kao preostalim aktuelnim serverima, a isto tako kada se novi server doda u grupu, load balanser automatski kreće da redirektuje zahteve ka njemu.

Znači load balanseri:

- Distribuiraju klijentske zahteve efektivno ka različitim serverima
- Obezbeđuju visoku raspoloživost i pouzdanost slanjem zahteva isključivo ka onim serverima koji su online
- Obezbeđuju fleksibilnost po pitanju dodavanja i uklanjanja servera u skladu sa potrebama i zahtevima.

Neki algortmi load balansera:

- Round Robin – zahtevi se distribuiraju serverima sekvencijalno.
- Least Connections – zahtev se šalje serveru koji ima trentno najmanji broj konekcija ka klijentima, uzimajući u obzir i kapacitet svakog servera.
- IP Hash – Ip adresa se koristi u hash funkciji koja odredjuje server kome će zahtev biti prosleđen

S obzirom da naša aplikacija obavlja rezrevacije za korisnika, kao i da mu omogućava prikaz ličnih informacija i interakciju sa drugim korisnicima, očuvanje sesije nam je od izuzetnog značaja, kako ne bi došlo do ugrožavanja transakcija, te bismo primenili kocept koji load balanseri pruzaju *session persistance.* To znači da bi svi zahtevi od strane jednog klijenta bili upućeni jednom serveru.

Caches

Keševi su najrasprostranjenije rešenje za ubrzanje opsluživanja zahteva, odnosno vraćanja traženih podataka, koji ni na koji način ne ugrožavaju platformu, te bismo se opredelili i za ovu tehniku ubrzanja performansi naše aplikacije.

Ideja je da će upit koji je bio sproveden u prošlosti sa visokom frekvencijom verovatno opet biti sproveden, te je ekonomično sačuvati ga, odnosno staviti ga u mali repozitorijum namenjen frekventnim zahtevima kako bi se smanjlo vreme čekanja.

U našoj aplikaciji ovo bi bilo pogodno na primer u slučaju da korisnik planira putovanje pa često ulazi da vidi letove određenog datuma ili hotele i rent a car servise u željenom mestu.

- Keš može biti ubačen u svaki node koji opslužuje zahteve kako bi se povratak podataka učinio skoro pa istovremenim. Porast broja nodova aplikacije proporcionalno vodi povećanju keš memorije.
- Može se ubaciti i globalni keš za sve nodove kako bi se izbeli brojni promašaji.

- Distribuirani keš obeybeđuje da se podacio distribuiraju između nodova, što su bliži korisniku to je odgovor brži**.**

Replikacija baze podataka

Repliciranje baze podataka je održavanje kopije baze podataka na više servera. To održavanje se ne radi tako što se kopiraju svi podaci s jednog servera na ostale, nego se promene koje su nastala na „master“ serveru izvršavaju i na „slave“ serverima. „Slave“ server ne mora nužno imati iste podatke kao „master“ server. To zavisi od toga što želimo replicirati.

Serveri u replikaciji mogu imati dve uloge. Glavni server se zove „master“ i bez njega nije moguće ostvariti replikaciju. Podređeni server se zove „slave“ i nije ga potrebno obavezno imati u replikaciji. Znači, replikacija se može ostvariti i s više „master“ servera koji su međusobno povezani bez prisustva „slave“ servera. „Master“ serveri omogućuju održavanje baze podataka, a „slave“ serveri repliciraju podatke s „master“ servera i obično ne omogućuju nikakve promene podataka.

Replikaciju možemo koristiti u razne svrhe. Kada se dogodi elementarna nepogoda na lokaciji gde je naš server s podacima, tada je dobro imati server s istim podacima na drugoj lokaciji. To je od izuzetnog značaja za našu aplikaciju, koja evidentira rezervacije letova, hotela i vozila i bila bi enormna šteta kada bi se one izgubile jer prosto moramo imati informacije prema kojim osobama imamo obaveze. Također je dobro imati odvojene servere, na raznim lokacijama, ako se rade velika učitavanja sa servera s različitih lokacija, što je opet slučaj kod naše aplikacije koja je od interesa korisnicima širom sveta. Kada trebamo nadograditi naš sistem i testirati je li sve radi ispravno, tada je bolje to raditi na pomoćnom „slave“ serveru a s obzirom da bi aplikacija 3Dots bila naša prva komercijalna aplikacija, ovo bi bilo pametno. Kada su kapaciteti čitanja preopterećeni, tada se čitanja sa servera mogu preusmjeriti na više „slave“ servera, što je odlično za opsluživanje neregistrovanih korisnika koji imaju samo pravo pregleda naših usluga.

CDN

Brzina učitavanja jedan je od najkritičnijih kriterijuma koji mogu napraviti ili slomiti uspeh web stranice. Mreža za isporuku sadržaja (CDN) postaje sve popularnija tehnika za poboljšanje brzine učitavanja web stranice, više od 66% od top 10,000 web stranica koristi mrežu za isporuku sadržaja.

Ona korisnicima pruža web stranice i drugi sadržaj na temelju njihovih geografskih lokacija. Drugim riječima, pomaže smanjiti vrjeme udaljenog servera da odgovori na podatke koje traže krajnji korisnici.

Obično se hvata statički sadržaj pohranjen na najbližem mogućem serveru u odnosu na geografski položaj korisnika. Kako se udaljenost kojom putuju podaci smanjuje, vrijeme isporuke (ili brzina učitavanja) poboljšava se.

S obzirom da je naša aplkacija turistički orijentisana i da je namenjena korisnicima širom sveta, ova tehnika bi značajno poboljšala njene performanse.

.