SKALABILNOST

Skalabilnost predstavlja mogućnost prilagođavanja kapaciteta sistema da ekonomično ispuni zahteve. Skalabilnost često podrazumeva mogućnost rukovanja sa mnogo korisnika, klijenata, podataka, transakcija.

Ukoliko aplikaiju koristi velik broj korisnika dolazi do rukavanja sa velikim brojem podataka, sto predstavlja veliki izazov za samu aplikaciju. Pojavljuje se veliki broj entiteta, I samim tim sto aplikaciju koristi puno ljudi dovodi do usporavanja iste.

Takodje problem je I konkurentnost. Konkurentnost se meri time koliko klijenata sistema može da bude opsluženo istovremeno, bez narušavanja iskustva korisnika aplikacije. Postoje aplikacije I server koji imaju tacno odredjeno koliko niti moze u jednom trenutnu da postoji. Ukoliko se desi da ima previse korisnika narusava se rad sa aplikacijom.

Na skalabilnost utice I nivo interakcije korisnika sa aplikacijom.

Skaliranje se svodi na horizontalno I vertikalno. Vertikalno skaliranje predstavlja ostvarenje nadogradnjom hardvera i/ili mrežnom komunikacijom. Često je najjednostavnije rešenje za kratkoročnu skalabilnost, jer ne zahteva arhitektonske promene u aplikaciji. Vertikalno skaliranje ima svoju granicu dokle se koristi, kasnije horizontalno sklaliranje prevazilazi problem vertikalnog skaliranja. Horizontalno skaliranje je skuplje I upravo zbog toga zahteva vise posla.

LoadBalancing je jedno od resenja koje bi poboljsalo pitanje skalabilnosti. Posto aplikacija koristi HTTP protocol koji je stateless, tada podate ne vezujemo za sesiju I zato je moguce koristiti loadBalancing. Ovde imamo vise server koji izvrsavaju zahteve klijenta I jedan server koji ima svoju adresu I port koji gadjamo. On je zaduzen da dalje rasporedjuje te zahteve na neki od ostalih servera. SpringSecurity nam pomocu tokena obezbedjuje da prepoznamo korisnika, I obezbedjuje nam bezbednost aplikacije. Ukoliko se neki server srusi moguce je preusmeriti zahtev na neki od ostalih server. Load Balanser koristi razlicite algoritme, ali se zahtev obicno salje onom serveru koji je najmanje opterecen.

Veliki broj podataka u bazi trosi puno vremena serveru da dodje do podatka koji mu treba. Jedan od optimalnih nacina da se lakse dolazi do podataka jeste da se podaci grupisu na odredjene nacine. Mogu da se grupisu po odredjenim poljima. Na primer jedan vid grupisanja podataka jeste da se na nivou nase apliacije Doktori podele na osnovu toga kojoj klinici pripadaju. Takodje, jedan vid grupisanja jeste grupisanje po stranim kljucevima. Moguce je grupisanje po kolonama I po vrstama.

Moguce je izvrsiti replikaciju baze kako bi se lakse dolazilo do podatka koji nam treba, kako ne bismo morali da idemo kroz celu bazu I trazimo podatak. Znali bismo tacno gde treba da trazimo taj odredjeni podatak.

Kesiranje pomaze na primer ukoliko imamo neki cest upit u bazu I kako ne bismo morali svaki put pri tom cestom upitu da trazmo iste podatke. Ukoliko su ti podaci kesirani smanjujemo vreme serveru da ih opet trazi u bazi I izlistava sve podatke. Veliki broj upita u bazu takodje dovodi do usporavanja aplikacije.

Transakcije pomazu kako ne bi doslo do losih izmena u bazi, do redundanse podatka.

TAMARA JOVICIC

ALEKSANDRA URBAN

ZELJANA SIPOVAC