1. Dizajn šeme baze podataka

   Priložen je u folderu dokumentacija zbog preglednosti.

2. Predlog za particionisanje podataka

   Particionisanje podataka može znatno pomoći u smanjenju broja zahteva za pristup resursima. Tabele sa velikim brojem podataka kao što su svi entiteti za rezervisanje bi podelili horizontalnim particionisanjem na osnovu identifikatora. Vertikalno particionisanje za tabele entiteta ne bi imalo smisla jer se često pristupa većini podataka dok bi se horizontalnim particionisanjem znatno poboljšale performanse.

   Vertikalno particionisanje bi iskoristili na tabelama korisnika. Odlicno bi bilo vertikalno particionisati podatke vezane za prijavljivanje (poverljive podatke) kojima se najviše pristupa i na koje bi u toj particiji mogli primeniti I dodatne sigurnosne provere. Drugu particiju bi činili uobičajeni podaci vezani za korisnike koji se ne koriste često kao sto su ime, prezime, broj telefona, adresa I slično.

3. Predlog strategije za replikaciju baze I obezbedjivanje otpornosti na greške

   Pošto je glavna namena naše aplikacije rezervisanje vikendica, brodova I avantura očekuje se veliki broj write operacija. Medjutim u praksi ljudi će dosta više razgledati ponude na sajtu I neće napraviti rezervaciju. Takodje oni koji naprave rezervaciju pre toga će pregledati sve opcije I uporediti prednosti I mane. Zbog navedenih očekivanja predlažemo “Multimaster“ arhitekturu za replikaciju baze gde bi imali više master baza. Svaka od njih bi obradjivala update I insert upite I vršila propagaciju izmena ostalim replikacijama kako bi podaci bili konzistentni. Zbog očekivanja da će postojati znatno veći broj read operacija dodali bismo I nekoliko slave baza koje bi pomogle za read upite.

4. Predlog strategije za keširanje podataka

   Po default-u naša spring aplikacija koristi hibernate koji automatski obavlja L1 keširanje. Za L2 keširanje smo implementirali primer pomoću EhCache java baziranog keša. Pomoću keširanja se perfomanse aplikacije mogu znatno poboljšati npr. veoma popularne vikendice će imati veliki broj read zahteva u malim vremenskim razmacima. Svi entiteti sadrže veliki broj slika koje predstavljaju najzahtevnije resurse te bi njihovo čuvanje u keš memoriji znatno poboljšalo performanse. Predlažemo “Least recently used” strategiju pri kojoj se keširaju rezultati poslednjih upita.

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

Objekat	Prosecna veličina	Pretpostavka	Potrebni resursi
Klijent	0.4 kb	85% korisnika	40GB
Vlasnik	0.35 kb	15% korisnika	6GB
Entitet	0.25 kb	Vlasnik ima u proseku 2 entiteta	7.5GB
Rezervacija	0.1 kb	Entitet u proseku 5 rezervacija mesečno	900GB
Ocena	0.1 kb	50% rezervacija ocenjeno	450GB

   Glavni problem oko hardverskih resursa predstavljaju slike koje bi bez kompresovanja potencijalno zauzele 300TB (svaki korisnik 1 profilna slika po otprilike 3mb) za profilne slike korisnika I 900TB za slike entiteta (svaki entitet u proseku 6 slika po 5mb) .

6. Predlog strategije za postavljanje load balancera

   Load balanser koristimo za smanjenje opterećenja servera. Za našu aplikaciju bile bi pogodne round robin I least connection strategije. U zavisnosti od toga da li su serveri istih konfiguracija potrebno je koristiti weighted strategije. Predlažemo least connection strategiju zbog velikog broja korisnika.

7. Predlog koje operacije treba nadgledati u cilju poboljšanja sistema

   Da bi poboljšali rad sistema moramo nadgledati koje operacije se najčešće koriste I utiču na performanse.

   a) Potrebno je nadgledati operaciju pretrage vikendica/brodova/avantura jer predstavljaju skupu operaciju zbog pretrage po pocetnom I krajnjem datumu I vremenu, kapacitetu. Ova operacija se izuzetno često koristi I njeni rezultati o zauzetosti I dostupnosti moraju biti tačni.

   b) Potrebno je nadgledati operacije rezervacije I rezervacije entiteta na akciji jer predstavljaju suštinu rada ove aplikacije. Pravljenje rezervacije predstavlja transakcionu operaciju što je čini skupljom od drugih.

   c) Potrebno je nadgledati operacije dobavljanja entiteta , naročito dobavljanje njihovih statičnih resursa (slika) kako bi se poboljšalo keširanje što bi dosta pozitivno uticalo na performanse rada aplikacije s obzirom da su slike najskuplji resursi s kojima naša aplikacija radi.

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