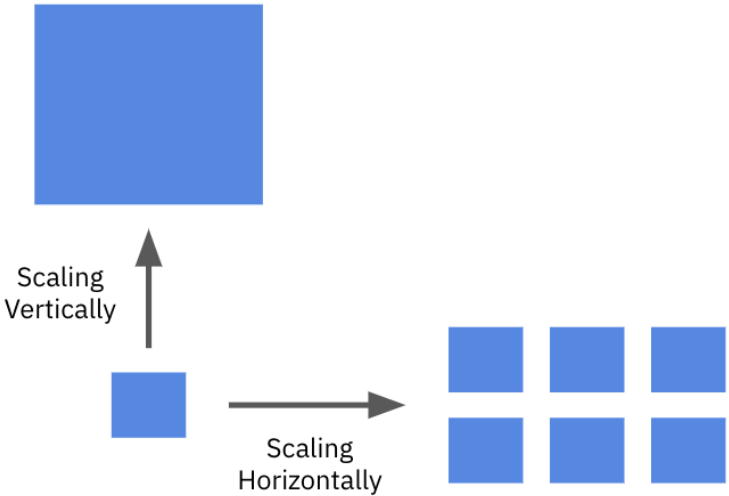
NoSQL baze podataka

Predavanje 2: Skaliranje, Key-value

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Uvod

- Skaliranje je proces u kom omogućavamo našoj aplikaciji/sistmeu da opslužuje rastući broj korisnika
- Kada se priča o skaliranju obično se misli na jedan od dva načina skaliranja koja su bitno različita
  - Vertikalno skaliranje je proces u kom se *uzima jača mašina*, tj. više resursa i u ovom principu ne postoji redudancija ako sistem padne naša aplikacija je nedostupna — preferenrani način skaliranja relacionih baza
  - Horizontalno skaliranje je proces skaliranja za *veće* aplikacije, gde se aplikacija izvršava na više mašina a zahtevi se balansiraju. Ako jedna mašina padne, uvek imamo druge koje mogu da prihvate zahtev — preferiran način skaliranja u cloud sistemima i NoSQL bazama



(Horizontalno i Vertikalno skaliranje)

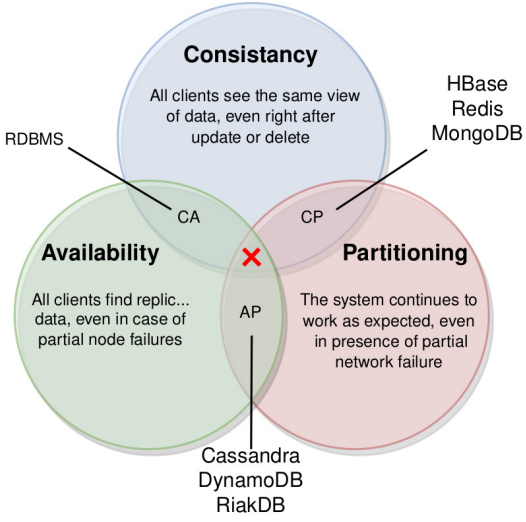
Replikacija sadržaja i koncenzus

- Pošto sa NoSQL bazama uglavnom pričamo o vertikalnom skaliranju, moramo videti kako se replicira sadržaj
- Ideja iza ovog mehanizma je rlativno jednostavna za razumeti, ali nije tako jednostavna uvek implementirati korektno
- Pošto nam se baza nalazi na *nekoliko čvorova* podaci su rastavljeni i ne nalaze se na jednom mestu
- Težimo da podatke grupišemo na nekakav način, uglavnom po opsegu ključeva
- Ali želimo da se podaci nalaze na nekoliko mesta – čvorovi mogu da nestanu sa mreže iz više razloga

- Da bi povećali dostupnost sistema, otpornost na greške repliciraćemo sadržaj na još n čvorova u sistemu
- Broj replika u sistmeu je obično *neparan*!
- Neparan je zato što nam treba većina od ukupnog broja čvorova gde se podatak čuva
- To znači da prilikmo zapisa podatka na čvor, želimo da ga repliciramo na još n mesta da bi obezbedili prethodne garancije
- Ovde može doći do konflikta u podacima, ali za to postoje tehnike kako se to rešava i ovaj posao nije nužno jednostavan

CAP teorema – podsetnik

- Ova teorema kaze da ne možemo zadovoljiti sve tri osobine u isto veme:
  - Konzistentnost
  - Dostupnost
  - Particioniranje
- Uvek možemo da dobijemo najviše dve od tri osobine
- Odluka se donosi prema potrebama aplikacije – najbolji model za dati problem
- Particioniranje nije moguće izbeći u distribuniram sistemima


Consistancy
All clients see the same view of data, even right after update or delete
HBase
Redis
MongoDB
RDBMS
CA
CP
Availability
All clients find replic... data, even in case of partial node failures
AP
Partitioning
The system continues to work as expected, even in presence of partial network failure
Cassandra
DynamoDB
RiakDB

(CAP teorema)

Uvod

- Ovaj model je jedan od najstarijih modelima korišćen je dosta i u samom UNIX operativnom OS-u
- Sve što nam ovaj model omogućava jeste da pod nekakvim ključem (uglavnom) tipa string, sačuvamo proizvoljan niz bajtova
- Isto tako, ako tražite vrednost pod odgovarajućim ključem dobićete nazad kompletan podatak koji se čuva pod tim ključem kao niz bajtova
- Izuzetno jednostavan model, izuzeno koristan model, izuzetno moćan model za razne alate i servise a vrlo sličan strukturi podataka *hash map*

- Model podržava par funkcionanosti kao osnovni skup operacija:
  - GET, vraća vrednost po ključu
  - PUT, zapisuje vrednost po ključu
  - DELETE, briže vrednsot po ključu
- Brisanje je obično implementirano kao novo dodavanje, gde se neki ključ markira za obrisan
- Izmena je zapravo novo dodavnje na isti ključ
- Prilikom procesa klompakcije, obrisane vrednosti i izmenjee vrednosti će biti ukljonene
- Uglavnom se gleda vremenska odrednica, šta se desilo kasnije – *LAST WRITE WINS*

Key-value i CAP

- Key-value model uglavnom preferira dostupnost naspram konzistencije
- Ovde nemamo toliko problema sa konzistenciom podataka
- Agregat sa kojim radimo je svakako enkapsuliran u celoj vrednosti!!
- AKo pak želimo da rastavimo podatak na nekoliko ključeva onda možemo imati potencijalan problem
- Neke baze dopuštaju i primitivan mehanizam transakcije u smislu da se agregat podataka, čak i rastavljen može ceo zapisati ili odbaciti

TTL

- Ovaj model uglavnom podržava i *Time to Live (TTL)* mehanizam
- Odnosno vremenski čuva podatak
- Ovo može biti zgodno za implementaciju funkcionalnosti vaših aplikacija
- Ova operacija nije tako jednostavna za implementriati u relacionij bazi, i obično zahteva upotrebu još minimalno jednog alata
- Cela ideja iza ovog tipa podataka je da on bude aktivan neki fiksan vremenski interval T_d, nakon čega biva obrisan

Prefix scan

- Key-Value model podržava i *prefix scan*
- Pretraga svih ključeva na osnovu prefixa — nešto slično select naredbi
- Koristeći ovu operaciju možete jednostavno vratiti sve podatke korisnika
- Naravno, moramo voditi računa kako dizajniramo ključeve da bi podržali ovu opciju
- Na primer:
  - key - user123_name, value - Miloš
  - key - user123_lastname, value - Simić
  - Ako radimo prefix skan sa user123, dobijamo nazad oba podatka!

Range scan

- Key-Value model podržava i *range scan*
- Pretraga svih ključeva u nekom opsegu oblika $[k_1, k_2]$
- Može biti zgodno kada želmo da dobijemo podskup ključeva koji nas zanimaju
- Ili da dobijemo veće detalje o nekom specifičnom delu sistema
- Treba voditi računa kako dizajniramo ključeve

Paginacija sadržaja

- Ako radimo range san ili prefix scan, podataka može da bude relativno puno
- Ne želimo da vraćamo sve podatke nazad korisniku preko mreže
- Treba nekako da omogućimo mehanizam *LIMIT* n koji podržava relaciona baza
- Želimo da uvek vratimo deo podataka dok ne stignemo do kraja skupa
- Korisnik zahteva stranicu prema nekakvom identifikatoru ili indeksu
- Uvek se vraća stranica po stranica

- Ovi modeli baza podataka uglavnom implementiraju mehanizam paginacije podataka
- To znači da prilikmo zahtevanja sadržaja, možemo ceo skup da podelimo na n stranica finse dužine
- podatke vraćamo po stranicama
- Trba voditi računa da može da se desi na nisu sve stranice popunjene i to je ok
- Druga stvar oko koje se mora voditi račina jeste da skup može da se proširi ili smanji u medjuvremnu
- treba ispratiti ove promene i korisniku vratiti samo ispravne informacije

Watcher

- Ovaj model realtivno jednostavno omogućava nadgledanje sadržaja
- Pod sadržajem se isključivo misli na ključ!
- Ova funkcionlanost može biti jako korisna
- Cela ideja je vrlo prosta – prijavite se da nadgledate ključ ili nekakv prefix ili range i kada se podatak izmeni vi budete notificirani
- Vrlo je zgodno kada imate komponenataln sistem, i želite da implementirate propagiranje informacija kroz komponente kada do se desi promena

Transakcije

- Transkacije su podržane, ali ne kao u relacionim bazama
- Transkacije su na nivou agregata zapisa, što je u ovom slučaju vrednsot za jedan ključ
- Ove baze garantuju da će vrednsot biti zapisana pod nekim ključem bez obzira koliko je vrednost komplikovan podataka
- Ovo je reltinvo lako, zat što je vrednost niz bajtova...
- Neke baze ovog tipa dopuštaju i primitivan oblik prave transkacije ako zapisujemo ili menjamo nekoliko ključeva

- Ovaj model može i da se koristi za implementaciju distriobuiranih transkacija
- DISTRIBUIRANE TRANSKACIJE IZBEGAVATI!
- Ako imamo više učeniska *SAGA* patern je mnogo jednostavniaj opcija
- Često se i koristi kao loking mehanizam i kao zapis ko ima *lock* u sistemu
- Dosta se koristi za sesije, shopping cart, kao skaldište za konfiguracije
- Iako jednostavan key-value model je jako korišćen u realnim aplikacijama

Streaming i Big Data

- Ovaj model se može lepo koristiti i za Big Data aplikacije i pre svega za streaming tipove aplikacija
- Stremaing podataka je način rada, gde podaci kontinuirnai stižu kroz vreme
- Od sistema se očekuje da se ov ipodaci nekako obrade, agregiraju i skladišti samo šta je bitno
- Obzirom da vrednost može biti bilo šta kao niz bajtova, možemo serijalizovati i napredne strukture podataka
- Ovi še često i radi – svi zapisi su prvo u memoriju što je dosta brzo

- Stoga veliki broj ovih baza podržava i probabilsitičke tipove podataka
- Ovi tpovi podataka mogu da skladište praktično beskonačnu količinu zapisa u fiksnu količinu memorije
- Naravno, precistno trpi!
- Stoga kod ovih struktura radimo procenu, a ne dobijemo egzaktnu vrednost nazad
- Ovo je jako korisno za streaming aplikcije gde real-toime nešto moramo da proračunamo, a da ne zauzmemo sve resurse ikada dostupne

Dodatni materijali

- Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement
- Dynamo: Amazon's highly available key-value store

Pitanja

Pitanja :) ?