2. Key/Value Stores Redis

Friday, 6 November 2020 10:49

Key/Value Stores
- Key za pretrazivanje I blob za podatke. Kljuc moze da bude blob al ne kompleksan zbog tezine pretrage
- Bazu ne zanima sta je blob samo ga salje kad se pozove, blob je bilo sta
- Jedan kljuc jedan podataka
- Nema duplikata kljuceva, brzo

Prednosti:
- Efikasna pretraga po kljucu (predvidljive preformanse)
- Jednostavna distribucija po klasteru - jednostavna fragmentacija prilikom upisa.
  - Podeli se tabela po horizontali I vrednosti se grupisu po klasteru
  - Ako koristimo kao kljuc timestamp jedan mesec jedan klaster drugi mesec drugi klaster
- Fkejsubilna sema - struktura podataka moze da se razlikuje I mora da podrzi sve vrste podataka u sistemu.
  - Struktura podataka se moze razlikovati od kljuca do kljuca
  - Kad promenimo strukturu sta sa starijim podacima?
- Memorijska efikasnost - cuvaju podatke koji postoje. Za null vrednosti se pamte al efikasno da ne zauzima prostor
- Nema problem object-relational impedace mismatch - zbog bloba sve moze da se cuva.
  - Vrati se blob I app radi sta radi sa time, bazu ne zanima sta je blob
- Cita se iz relacione baze upise u key/value sistem I cita se posle iz key/value ne iz relacione baze

Lose strane
- Nema kompleksnost upita- jer sve preko key se radi, ne pretrazuje se sastav bloba
- Nema stranih kljuceva
- Spoj podataka je zavisna od aplikacija
- Nema standardizacije

Redis - REmote DIctionalry Server
- Ima kljuc a blob je neka struktura podataka

- In-memory baza podataka - sve se cuva u memoriji, postize se ogromna brzina, gasenje baze gubi podatke al redis ima nacin cuvanja podataka.
  - Cuvanje podataka:
    - Snapshoting - cela slika periodicno se snimi na disk, problem jer moze da bude veliki pa traje
    - Apend-only File - kako se menja podatak u mem samo se te promene periodicno snime

  - Jer se cuva u memoriji pa tek kasnije se cuva u bazi moze da dodje do gubljenja podataka
  - Bitno nam je availability ne trajnost(ovo iznad)

- Implementiran kao jedan proces sa jednom niti (single traded process)
  - operacije su brze, zbog jedne niti sekvencijalno se rade naredbe, jedna nit jedan poso radi
  - Ako ima vise zadataka radi se po FIFO sistemu I izvrsava se jedno po jedno nema paralelizacije, zbog ovog nema zakljucavanja baze

- Tipovi podataka:
  - Binary safe kljuc - bilo koja binarna sekvenca, rec, br, slika itd. Max velicina je 512mb

    - Podatci mogu biti: Stringovi, Liste, Skupovi (Sets), Sortirani skupovi (Sorted sets), Bitmape (Bit arrays), Hyper LogLogs
      - Odabir tipa podataka zavisi od zahteva posla
    - Podaci max velicina 512mb
    - Prazan string je validan kljuc

    - Operacije:
      - Set - na osnovu kljuca se upisuju podaci, ako kljuc ne postoji kreira se u redisu
        - Kod:
        - > set key value
        - >set counter 100
      - Get - na osnovu kljuca se nalaze podaci, ako kljuc ne postoji dobija se null vrednost
        - >get counter
        - 100
      - INCR, INCRBY – atomična operacija za inkrementiranje vrednosti. Ne posotji mogućnost pojave race conditions.
        - >incr counter
        - (integer) 101
        - >incrby counter 50
        - (integer) 151
      - DECR, DECRBY – atomična operacija za dekrementiranje vrednosti.
      - GETSET – postavlja novu vrednost za zadati ključ i istovremeno vraća prethodnu vrednost ključa

MSET – postavlja vrednosti za veći broj ključeva
MGET – pribavlja vrednosti za veći broj ključeva

Podaci koji su obradjeni moze da se cuvaju u Redisu kao JSON I da se posle pribave gotovi

U kodu se pamti kao string, da bi matematicki radio mora (integer) pre imena da stavimo da se zna koja vrsta je podatak posle kod operacije
INCR pokusava da konvertuje string u integer pre inkrementacije

Implementirana kao lancana lista
Sa pokazivacem na pocetak I kraj liste
FIFO sortiranje liste
Dodavanje elementa je na pocetak ili kraj pa je efikasna operacija

Operacije za rad sa listama:
LPUSH/RPUSH – dodavanje novog elementa na početak/kraj liste. Podržavaju dodavanje većeg broj elemenata odjednom.
LPOP/RPOP – pribavlja prvi/poslednji element iz liste i istovremeno ga uklanja iz liste.
LRANGE – pribavlja opseg elemenata (podlistu elemenata) iz liste.
LTRIM – izabacuje iz liste sve elemente van definisanog opsega.
LLEN – vraća dužinu liste (broj elemenata)
LREM – briše element iz liste sa zadatim indeksom
LINSERT – ubacuje novi element u list ispred ili iza elemenata sa zadatim indeksom

>rpush mylist A		>rpush mylist A B C
(integer) 1		(integer) 3
>rpush mylist B		>rpop mylist
(integer) 2		"C"
>lpush mylist first		>rpop mylist
(integer) 3		"B"
>lrange mylist 0 -1		>rpop mylist
1) "first"		"A"
2) "A"		
3) "B"		

U lrange stavlja se 0 -1, pocni od nule I za -1 znaci idi do kraja

Redis set - neuredjenu kolekciju stringova, skup
Nema duplikata elemenata jer je SET
Elementi su neuredjeni
Operacije
SADD – dodavanje jednog ili većeg broja elemenata u skup
SISMEMBER – proverava se da li se zadata vrednost nalazi u skupu
SMEMEBERS – pribavlja sve vrednosti iz skupa
SREM – briše jednu ili veći broj vrednosti iz skupa
SRANDMEMBERS – vraća jednu ili više slučajnih vrednosti iz skupa
SPOP – vraća slučajnu vrednost iz skupa i istovremeno je briše iz skupa
SINTER, SUNION, SDIFF – operacije između skupova

>sadd myset 1 2 3		>sadd news :1000:tags 1 2 5 77	Ime :id:tags
(integer) 3		(integer) 4	
>smembers myset	Vraca sve		
1. 3 2. 1 3. 2		>sadd tag :1:news 1000 (integer) 1	
		>sadd tag :2:news 1000 (integer) 1	
>sismember myset 3		>sadd tag :5:news 1000 (integer) 1	
(integer) 1		>sadd tag :77:news 1000 (integer) 1	
>sismember myset 30			
(integer) 0		>smembers news :1000:tags	
		1. 5	

		2. 1 3. 77 4. 2	

Desna strana:

Nema kljuceva I referenci pa ovim tagovima mi uvodimo funkcionalnost trazenja kao da imamo kljucevge
Prva linija koda I sledece 4 rade istu stvar dodavanja tagova jer se svi isto zovu news I imaju isti ID

Redis HASH

Velicina Hash-a je 4 milijarde parova
Predstavljanje objekata

Operacije:

HSET/HGET – postavlja/probavljas string vrednost nekog od atributa unutar hash-a
HMSET/HMGET - postavlja/probavljas string vrednost većeg broj atributa unutar hash-a.
HLEN – vraća broj atributa koji su definisani u okviru hash strukture
HKEYS – vraća sve atribute koji postoje u hash strukturi
HGETALL – vraća sve atribut/vrednost parove koji posotje u okviru hash strukture

>hmset user :1000 username antirez birthyear 1997 verified 1
OK

>hget user :1000 username
"1977"
>hgetall user :1000
1) "username"
2) "antirez"
3) "birthyear"
4) "1997"
5) "verified"
6) "1"

Prva linija: user :1000 je kljuc za nalazenje mape(hasha)

posle su informacije na toj poziciji parovi kljuc vrednost (kljuc: username, vrednost: antirez)

Redis sortirani set

SCORE je vrednost koja je pridruzena svakoj svakom elementu.

A>B ako je A.score > B.score
A=B onda se vrsi leksikografsko poredjenje (poredjenje po stringu)

Jednaki su ako I score I leksikografsko poredjenje je isto

Operacije

ZADD – dodaje jedan ili više elemenata u sortirani skup ili menja njihov score ukoliko elementi već postoje
ZREM – briše jedan ili više elemenata iz sortiranog skupa
ZRANGE – vraća elemente is sortiranog skupa koji se nalaze u zadatom opsegu

```
> zadd hackers 1940 "Alan Kay"
(integer) 1
> zadd hackers 1957 "Sophie Wilson"
(integer 1)
> zadd hackers 1953 "Richard Stallman"
(integer) 1
> zadd hackers 1949 "Anita Borg"
(integer) 1
> zadd hackers 1965 "Yukihiro Matsumoto"
(integer) 1
> zadd hackers 1914 "Hedy Lamarr"
(integer) 1
> zadd hackers 1916 "Claude Shannon"
(integer) 1
> zadd hackers 1969 "Linus Torvalds"
(integer) 1
> zadd hackers 1912 "Alan Turing"
(integer) 1
```

```
> zrange hackers 0 -1
1) "Alan Turing"
2) "Hedy Lamarr"
3) "Claude Shannon"
4) "Alan Kay"
5) "Anita Borg"
6) "Richard Stallman"
7) "Sophie Wilson"
8) "Yukihiro Matsumoto"
9) "Linus Torvalds"
```

Lista ljudi sortirana (desno) po godini rodjenja

Redis bitmapa

Radi se o stringu pa je max velicina 512MB
Dobro za true/false info (yes no, on off)

Operacije:

SETBIT/GETBIT – postavlja/pribavlja vrednost određenog bita
BITOP – bit-wise operacija između različitih stringova
BITCOUNT – vraća broj bitova koji imaju vrednost 1
BITPOS – poziciju bita koji ima zadatu vrednost (0 ili 1)

```
> setbit key 10 1
(integer) 1
> getbit key 10
(integer) 1
> getbit key 11
(integer) 0
```

```
> setbit key 0 1
(integer) 0
> setbit key 100 1
(integer) 0
> bitcount key
(integer) 2
```

Vrednost na poziciji 10 je 1 pre upisa 1 pa se to vraca u odgovoru
Vrednost na pozicijama 0 I 100 je 0 pre upisa 1 pa se vraca 0 kao odgovor

ZA SVE TIPOVE VAZI

EXISTS – proverava da li specificirani klju postoji (vraća 1) ili ne postoji (vraća 0)
DEL – briše specificirani ključ i vrednost koja mu je pridružena. Vraća 1 ako je ključ obrisan ili 0 ako ključ nije postojao.
TYPE – vraća tip vrednosta koja je pridružena specificiranom ključu

```
> set mykey hello
OK
> exists mykey
(integer) 1
> del mykey
(integer) 1
> exists mykey
(integer) 0
```

```
> set mykey x
OK
> type mykey
string
> del mykey
(integer) 1
> type mykey
none
```

U redisu se uvodi TIMEOUT

Ako kazemo da podatak ima timeout 10sec on ce se obrisati posle 10 sec

EXPIRE – komanda kojom se definiše timeout
PERSIST – komanda koja briše definisani timeout

Korisno kod dobijanja sertifikata online. Umesto nonstop da pribavljamo sertifikat to cinimo samo jednom.
Onda sacuvamo u redis sertifikat I stavimo timeout 3600 sec (1h)

Redis transakcije

Transakcije su Atomicne, Konzistentne I izolovane (ACI) nema trajnost tj nije pun ACID (bez D)

Za razliku kod relacionih baza gde ako jedna operacija u transakciji otkaze cela transakcija se gubi
U redisu ako jedna operacija u transakciji otkaze nista se ne gubi samo se nastavi sa ostalim operacijama
Ukoliko dođe do greške prilikom baferovanja komandi Redis odbija izvršavanje transakcije
Korisno da se obezbedi izvrsavanje kao celina bez umetanje tudje operacije u transakciju jer je single threaded

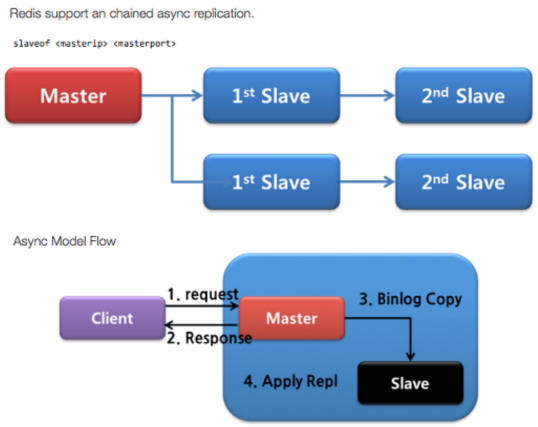
Komande:

MULTI omogućava grupisanje/baferovanje komandi koje će činiti transakciju
EXEC izvršava komande grupisane u transakciju

Redis Publish/Subscribe

PUBLISH - šalje poruke u kanal ne znajući da li postoje klijenti zainteresovani za prijem tih poruka.
SUBSCRIBE / UNSUBSCRIBE - prima podatke iz određenih kanala ne posedujući informaciju da li postoje klijenti koji šalju podatke u te kanale.

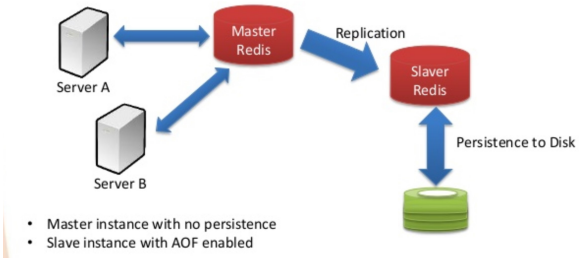
Master Slave mehanizam replikacije



Mehanizam sluzi za kreiranje pouzdanosti
Master - cita se I menjaju podaci
Slave - sinhronizuje se sa masterom I kopira sve podatke (vreme za kopiranje je konstantno)
Kopira se binarni log (opis operacija izvrsene nad masterom) I onda se salje na ostale slavove
Slave kopije mogu da se koriste za citanje podataka I tako da se skalira citanje podataka (samo u master se pise I menja)

Slave moze da ima nivoe (nivo 1, nivo 2 itd.)

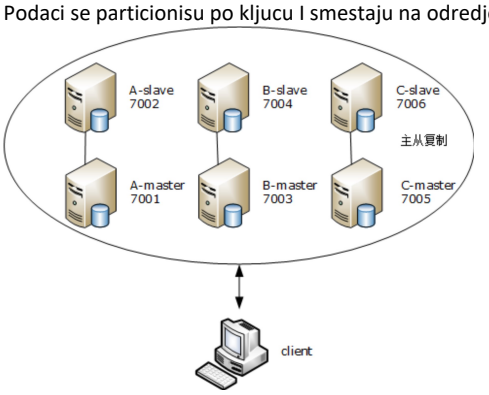
Redis - Data Persistence



Ovde master radi u memoriji I veoma je brz zbog toga, slave sluzi da obradjen log preuzme od mastera I stavi na disk

Redis cluster

Kad ima previse informacija pa jedna istanca redisa nije dovoljna, tj nema doboljno radne memorije
Podaci se particionisu po kljucu I smestaju na odredjeni node (par master slave)



Redis sentinel

Nema previse podataka, dovoljna je jedna instanca al zelimo pouzdano resenje
Master slave, ako otkaze master zamenjuje ga neki slave time sto postaje on master
Razlika od predhodnog je sto zamena se radi automatski kako ne bi bilo down time procesom monitoring
Jedan master I 2 slave min jer uvek treba da ima par jedan I u slucaju otkaza jednog ima 2 za zamenu

Redis primena:
Session cache - kesiramo info korisnika I akcija sesije
Full-page cache - kesiranje stranice
Red poruka (message queues)
Brojači (counters)

Leaderboards, scoreboards
Publish/Subscribe
Real-time analytics

Bilo sta gde treba brzina a ne cuvanje brzo (kesiraje)

Distribuirani locovi - gde treba da se obezbedi da dva cvora ne koriste isti resurs u isto vreme ili rade istu stvar u isto vreme

Dodaci za redis postoje
- Kao graf bazu
- Kao document bazu