Elektronski fakultet Niš

Seminarski rad iz predmeta

Sistemi za upravljanje bazama podataka

Tema:

Backup/restore **Redis** baze podataka

Student:

Milica Mladenović, 1061

Mentor:

Aleksandar Stanimirović

Sadržaj

Uvod

Redis sistem baza podataka je projekat otvorenog koda koji je nastao na osnovu ideje programera Salvatorea Sanfilippa 2009. godine. Trenutno se razvija u kompaniji RedisLabs i napisan je u programskom jeziku C.

Redis je akronim od **R**emote **D**ictionary **S**erver (*srp*. udaljeni rečnički server). Redis aplikacija se sastoji iz Redis servera i Redis klijenta. Redis server je zadužen za skladištenje podataka i čitavu logiku njihove obrade, dok Redis klijent predstavlja biblioteku implementiranu u programskom jeziku u kom je pisana aplikacija.

Redis server čuva podatke u primarnoj memoriji, što omogućava veoma brze operacije čitanja i pisanja u bazu podataka. S obzirom na to da se podaci koji se nalaze u primarnoj memoriji gube pri gašenju servera, Redis sadrži mehanizme za čuvanje podataka u sekundarnoj, trajnoj memoriji na neki od dva načina – čuvanjem svih podataka iz baze kada se zadati uslovi ispune (broj upisa u bazu, zakazano vreme itd.) ili čuvanjem svake od izvršenih naredbi nad bazom.

Redis je **NoSQL** sistem baza podataka, ili kako je često nazivaju skladište (*engl*. data store), koji organizuje podatke u ključ-vrednost parove. Ključ je niska koji jedinstveno određuje jedan zapis, dok vrednost može biti tipa: String, Lista, Skup, Uređeni skup, Heš i HyperLogLog.

Redis sistem baza podataka trenutno koristi veliki broj kompanija kao što su *Twitter, GitHub, StackOverflow, Pinterest, Instagram, Snapchat*. Može se koristiti kao baza podataka, kao sistem za keširanje podskupa podataka iz primarne baze podataka, keširanje sesije, kao implementacija redova zadataka, kao posrednik u slanju poruka itd.

Slika 1. *Redis logo*

Postojanost podataka u Redis-u

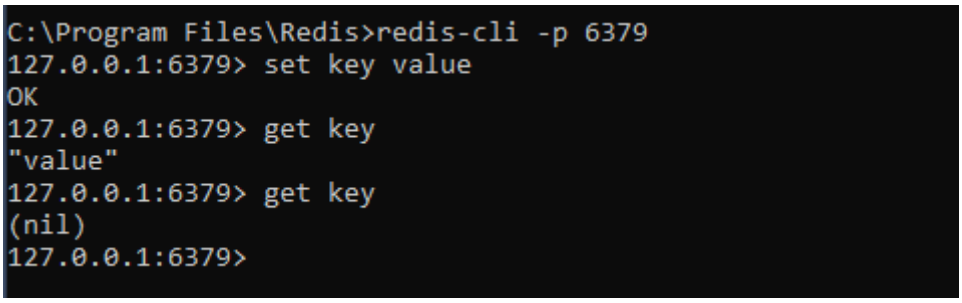
Jedna od glavnih karakteristika koja Redis izdvaja u odnosu na druge baze podataka je to što podatke čuva u RAM memoriji. Pristup RAM memoriji je znatno brži od pristupa sekundarnoj memoriji, pa su zato i operacije nad podacima znatno brže.

RAM memorija je nestalna i svi podaci bi nestali ukoliko bi server morao ponovo da se pokrene, ukoliko dođe do pada sistema, nestanka napajanja i slično. Redis nudi dva načina trajnog čuvanja podataka u sekundarnoj memoriji: pomoću kreiranja i čuvanja „snimka“ (*engl*. snapshotting) baze na disk u datoteku koja se obično naziva **R**edis **D**atabase **B**ackup (RDB) datoteka i pomoću upisivanja svih izvršenih naredbi pisanja u posebnu datoteku na disku (*engl*. **A**ppend **O**nly **F**ile, AOF).

Ukoliko postoji potreba, može se u potpunosti onemogućiti postojanost podataka i učiniti da oni postoje samo u trenutku kada je server pokrenut.

Primer:

Pretpostavimo da setujemo vrednost "value" za dati ključ "key". Nakon toga proverimo da li je podatak skladišten pomoću naredbe GET. Kada se uverimo da je podatak sačuvan u RAM memoriji, ručno ugasimo server. Zatim nakon ponovnog pokretanja servera, ponovo pozovemo naredbu GET za isti ključ. Rezultat će biti upravo gubitak podatka koji smo prethodno upisali (slika 2.).



Slika 2. *Primer gubitka podataka usled gašenja servera*

Dakle, može se koristiti jedan mehanizam, kombinacija oba ili nijedan od njih, zavisno od slučaja korišćenja. Osim što ti mehanizmi imaju za posledicu trajno čuvanje podataka, takođe se dobija mogućnost replikacije kopija baze na druge mašine, čime bi se dobilo i na performansama i na još većoj pouzdanosti u odnosu na onu koju dobijamo čuvanjem na samo jednoj mašini.

Najvažnija stvar koju treba shvatiti su različiti kompromisi između RDB i AOF mehanizama za postojanost podataka u bazi.

Prednosti RDB datoteke

- RDB datoteka čuva snimak Redis baze koji predstavlja sve podatke koji su smešteni u bazi do tog trenutka. Ove datoteke su savršene za izradu **backup**-ova, odnosno sigurnosnih kopija. Na primer, ukoliko želimo da svakog sata arhiviramo promene koje su se desile u bazi tokom prethodna 24 časa i da svakog dana čuvamo snimak baze u trajanju od 30 dana. To nam omogućava **restore**, odnosno da lako povratimo različite verzije skupa podataka u slučaju nekih iznenadnih i neplaniranih događaja.
- RDB datoteka je veoma dobra za oporavak od "katastrofe" jer predstavlja jedinstven i kompaktan fajl koji se može preneti u daleke centre podataka ili na Amazon S3 (najverovatnije u enkriptovanom obliku). Ona povećava performanse Redis sistema jer jedini posao koji Redis roditeljski proces treba da uradi kako bi istrajao je da fork-uje proces dete koje će uraditi sve ostalo. Roditeljska instanca nikada ne izvodi ulazno/izlazne operacije i slično. Takođe, RDB datoteka omogućava brže ponovno pokretanje sa velikom skupovima podataka u odnosu na AOF datoteku.

Nedostaci RDB datoteke

- RDB datoteka nije dobar izbor u situaciji kada treba smanjiti mogućnost gubitka podataka u slučaju da Redis prestane da radi, na primer nakon nestanka napajanja. Možemo konfigurisati različite tačke snimanja u kojima se generiše ova datoteka (na primer, nakon najmanje 5 minuta i 100 operacija upisa). Međutim, obično se kreira snimak baze na svakih 5 minuta ili više, tako da, ukoliko dođe do prestanka rada Redis servera bez ispravnog gašenja, može doći do gubitka podataka u poslednjim minutima.
- RDB često izvršava fork naredbu kako bi istrajao na disku koristeći podređeni proces, tj. proces dete. Fork() može oduzeti mnogo vremena ukoliko je skup podataka preveliki, a to može dovesti do toga da Redis server prestane da opslužuje klijente na nekoliko milisekundi ili čak na ceo sekund ukoliko je skup podataka veliki ili CPU performanse nisu dovoljno velike.

Prednosti AOF datoteke

- Korišćenje AOF datoteke je mnogo pouzdanije: postoje različite opcije sinhronizacije (*fsync*[1] operacija): svake sekunde, prilikom svake naredbe, ili bez podešavanja (operativni sistem bi na svoj način vršio sinhronizaciju).
- AOF datoteka je priložena datoteka tako da ne postoje bilo kakvi problemi sa korupcijom ukoliko dođe do prekida napajanja. Čak, ukoliko se desi da se datoteka završi polunapisanom naredbom iz nekog razloga (punog diska ili nečeg drugog), Redis-check-aof-tool može to lako da popravi.
- Redis je u stanju da automatski prepiše AOF datoteku u pozadinu ukoliko ona postane suviše velika. Prepisivanje je potpuno bezbedno jer, dok Redis nastavlja dodavanje staroj datoteci, nastaje potpuno nova datoteka uz minimalan skup operacija potrebnih za kreiranje trenutnog skupa podataka. Kada ona bude spremna, Redis se prebacuje na nju i u njoj započinje dodavanje operacija.
- AOF datoteka sadrži dnevnik svih operacija, jednu za drugom, u obliku koji se može lako razumeti. Ova datoteka se čak može i lako eksportovati. Na primer, čak iako se isprati sve vezano za neku grešku korišćenjem FLUSHALL naredbe, ukoliko u međuvremenu nije izvršeno prepisivanje dnevnika, i dalje je moguće sačuvati skup podataka tako što će se zaustaviti server, ukloniti poslednja naredba i ponovo pokrenuti Redis server.

Nedostaci AOF datoteke

- AOF datoteke su obično veće od ekvivalentnih RDB datoteka za isti skup podataka.
- AOF može biti sporiji od RDB-a u zavisnosti od usvojenog načina sinhronizacije. Generalno, podešavanjem fsync naredbe na svaku sekundu, performanse su i dalje veoma velike, a kad je fsync onemogućen, AOF bi trebalo da bude jednako brz kao i RDB, čak i pod velikim opterećenjem. Ipak,

[1] **fsync()** - prenosi sve modifikovane osnovne podatke datoteke na disk ili neki drugi uređaj za trajno skladištenje tako da se sve izmenjene informacije mogu povratiti čak i nakon pada ili ponovnog pokretanja sistema. Ova funckija je blokirajuća (sve dok uređaj ne prijavi da je prenos podataka završen).

RDB je u mogućnosti da pruži više garancija u vezi sa maksimalnim kašnjenjem, čak i u slučaju velikog opterećenja pisanjem.

- U prošlosti su postojale retke greške u specifičnim naredbama, što je uzrokovalo da AOF ne reprodukuje potpuno isti skup podataka prilikom ponovnog učitavanja. Sa druge strane, ove greške su gotovo nemoguće uz postojanje RDB datoteke. Naime, AOF datoteka u Redis-u deluje tako što postepeno ažurira postojeće stanje, kao što rade i MySQL ili MongoDB, dok RDB datoteka stvara snimke baze iznova i iznova što je konceptualno robusnije. Ipak, treba imati na umu da svaki put kada Redis prepiše AOF datoteku, ona se ponovo kreira od nule, počevši od stvarnih podataka sadržanih u skupu podataka, čineći otpor bagovima koji je veći u poređenju sa uvek priloženom AOF datotekom. Takođe, ne postoji nijedan izveštaj korisnika o korupciji AOF-a otkrivenoj u stvarnom svetu.

Opšta preporuka je da treba koristiti obe metode za postojanost podataka kako bi stepen sigurnosti podataka bio uporediv sa onim što može pružiti PostgreSQL. Ukoliko je nekome mnogo stalo do podataka, a ipak može da podnese gubitak podataka na nekoliko minuta, jednostavno može koristiti RDB datoteku. Veliki je broj korisnika koji koriste AOF datoteku, iako je mnogi potcenjuju jer je posedovanje snimka baze s vremena na vreme odlična ideja za pravljenje backup-a baze podataka i za brže ponovno pokretanje. U nastavku teksta slede dodatne informacije i konkretni primeri za ove dve metode postojanosti podataka.

Snimak baze

Snimkom baze u Redisu pravi se datoteka koji predstavlja podatke smeštene u bazi do tog trenutka. Po default-u, Redis čuva ove snimke na disku, u binarnu datoteku zvanu **dump.rdb**. Redis se može konfigurisati tako da čuva skup podataka na svakih N sekundi ako postoji najmanje M promena u skupu podataka, ili se jednostavno ručno mogu pozvati odgovarajuće naredbe. Ova strategija je poznata kao **snimak baze** (*engl.* snapshotting). Naime, postoji više načina izvođenja snimka baze:

- Pomoću naredbe SAVE koja je blokirajuća i stopira sve upite ka Redisu dok se snimanje ne završi.

```
SAVE broj_sekundi broj_promena
```

- Pomoću naredbe BGSAVE (background save) koja ne zaustavlja rad servera prilikom snimanja podataka. Ona kreira dete proces (fork) koji se bavi snimanjem podataka, dok roditelj proces sve vreme prima naredbe od klijenata i izvršava ih.

Naredba SAVE, sa argumentima broj_sekundi i broj_promena, kaže da će se čuvanje trenutnog stanja baze pokrenuti automatski na svakih broj_sekundi ukoliko se u tom intervalu desio broj_promena operacija pisanja od poslednje uspešno izvršene naredbe SAVE. U datoteci koja sadrži podešavanja Redisa može biti navedeno više takvih naredbi istovremeno, i čim se prva od njih ispuni, pokreće se proces pisanja datoteke.

Koliko god često podešavali iniciranje čuvanja snimka skladišta, nije zagarantovano da neće biti izgubljenih podataka. Ukoliko dođe do stopiranja Redis servera, sve promene nad postojećim, i svi novokreirani podaci nakon poslednjeg snimka će biti izgubljeni. Još jedna mana ovog mehanizma je što prilikom BGSAVE naredbe kreiranje procesa deteta može za cenu imati potpuno stopiranje sistema ukoliko imamo posla sa velikom količinom podataka koja ne ostavlja puno memorije za dodatno kreiranje procesa deteta. U tom slučaju, zbog bitke za resurse, sam proces čuvanja snimka trenutnog stanja Redis skladišta traje duže, i dolazi do značajnog pogoršavanja performansi sistema. Kako bi se to izbeglo, može se koristiti varijanta sa SAVE blokirajućom naredbom, ili se može potpuno isključiti automatsko iniciranje snimka i pokretati se ručno onda kada je najmanje frekventno korišćenje baze.

Primer korišćenja SAVE blokirajuće naredbe

U konfiguracioni fajl se doda linija kojom se definiše da se na deset sekundi proverava da li su se, od poslednje uspešno izvršene naredbe SAVE, desile četiri promene (slika 3.).

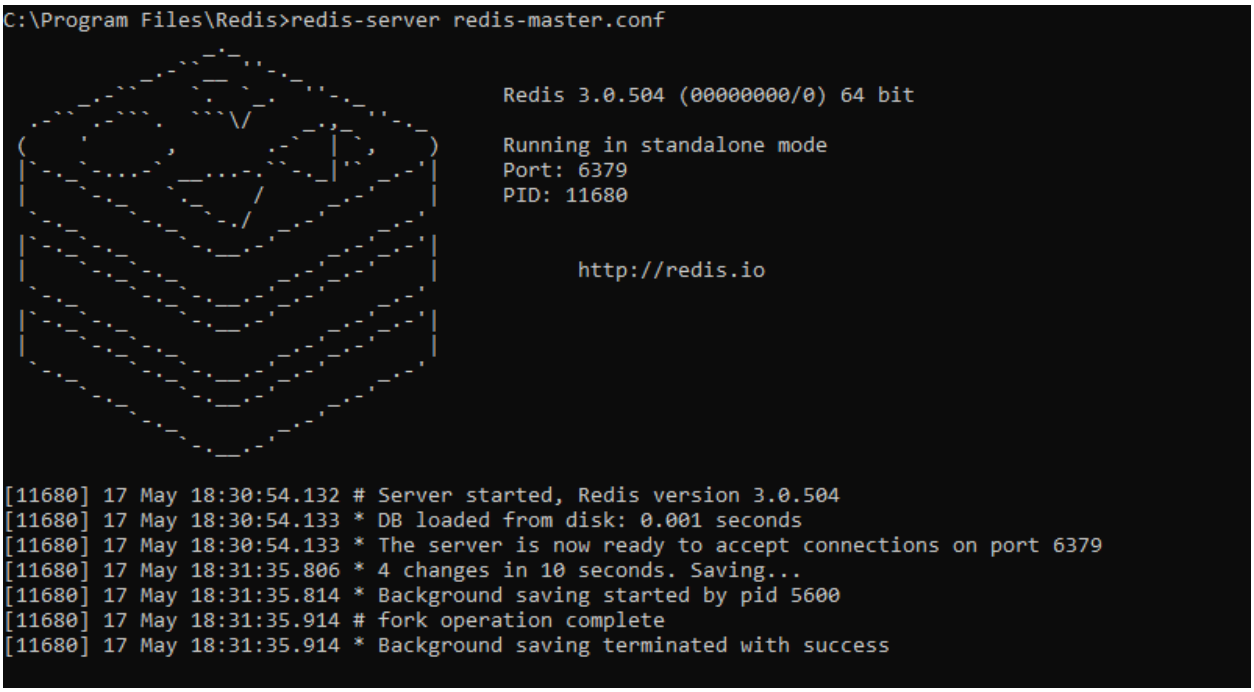
```
116 #
117 # Save the DB on disk:
118 #
119 #   save <seconds> <changes>
120 #
121 #   Will save the DB if both the given number of seconds and the given
122 #   number of write operations against the DB occurred.
123 #
124 #   In the example below the behaviour will be to save:
125 #   after 900 sec (15 min) if at least 1 key changed
126 #   after 300 sec (5 min) if at least 10 keys changed
127 #   after 60 sec if at least 10000 keys changed
128 #
129 #   Note: you can disable saving completely by commenting out all "save" lines.
130 #
131 #   It is also possible to remove all the previously configured save
132 #   points by adding a save directive with a single empty string argument
133 #   like in the following example:
134 #
135 #   save ""
136
137 save 10 4
138
139 # By default Redis will stop accepting writes if RDB snapshots are enabled
140 # (at least one save point) and the latest background save failed.
141 # This will make the user aware (in a hard way) that data is not persisting
142 # on disk properly, otherwise chances are that no one will notice and some
143 # disaster will happen.
144 #
```

Slika 3. *Konfiguracioni fajl sa SAVE naredbom*

Nakon 4 operacije upisa u bazu (slika 4.), pri prvom narednom pokušaju sinhronizacije (kraj n-tog intervala od 10 sekundi), desiće se uspešna sinhronizacija, odnosno snimak baze u fajl *dump.rdb* (slika 5.).

```
C:\Program Files\Redis>redis-cli -p 6379
127.0.0.1:6379> set key1 value1
OK
127.0.0.1:6379> set key2 value2
OK
127.0.0.1:6379> set key3 value3
OK
127.0.0.1:6379> set key4 value4
OK
127.0.0.1:6379> get key1
"value1"
127.0.0.1:6379>
```

Slika 4. *Naredbe upisa (SET)*



Slika 5. *Uspešan poziv naredbe SAVE*

Sledeća naredba upisa vrši upis podatka u RAM memoriju, gde će podatak biti siguran sve do trenutka dok ne dođe do događaja koji će nasilno prekinuti rad servera (na primer, nestanak struje). Nestanak struje je u ovom slučaju simuliran ručnim gašenjem procesa koji izvršava server (kombinacijom komandi *Ctrl* + *Alt* + *Delete* se otvara Task Manager pomoću kog prekidamo proces).

Prvom naredbom GET, za ključ *key5*, koja se poziva pre gašenja servera, zahteva se podatak iz RAM memorije i uspešno se pribavlja (slika 6.).

Drugom naredbom GET, za ključ *key5*, koja se poziva nakon gašenja i ponovnog pokretanja servera, ne dobijamo podatak jer je on izgubljen usled gašenja servera zbog toga što je upis izvršen između dve sinhronizacije (druga se nije još dogodila).

Sledeća naredba GET, za ključ *key1*, uspešno pribavlja podatak, i nakon pada servera, zato što je podatak prilikom sinhronizacije upisan u stalnu memoriju (u fajl *dump.rdb* odakle je i pribavljen).

Slika 6. *Primer gubitka podatka koji još uvek nije upisan u stalnu memoriju*

Append-only (AOF) datoteka

Metod snimanja baze nije izdržljiv i pouzdan. Ako se računar na kome se pokreće Redis server zaustavi, dođe do prekida napajanja ili se slučajno "ubije" instanca, poslednji podaci upisani u Redis bazu će biti izgubljeni. Iako ovo možda nije veliki problem za neke aplikacije, postoje slučajevi gde je potrebna potpuna trajnost i u tim slučajevima Redis nije bio pouzdana opcija.

AOF datoteka je alternativna, potpuno-trajna strategija za Redis bazu podataka. Postala je dostupna u verziji 1.1. Ona predstavlja mehanizam upisivanja u datoteku svake od naredbi pisanja u bazu, redosledom kojim se izvršavaju. Time se kompletna baza može rekreirati tako što se izvrše naredbe iz datoteke. U konfiguracionom fajlu se može omogućiti korišćenje ove datoteke na sledeći način:

```
appendonly yes
```

Od tog trenutka pa na dalje, kada Redis dobije naredbu da promeni skup podataka (na primer, naredba SET), ta naredba će se pojaviti u AOF datoteci. Kada se Redis ponovo pokrene, on će ponovo reprodukovati AOF kako bi obnovio stanje.

AOF datoteka postaje sve veća i veća kako se povećava broj operacija upisa. Na primer, ako postoji neki brojač koji se povećava 100 puta, na kraju će u skupu

podataka postojati samo jedan ključ koji sadrži konačnu vrednost, ali će biti 100 unosa u AOF datoteku, od kojih 99 nije potrebno za obnovu trenutnog stanja. Dakle, Redis poseduje zanimljivu karakteristiku: može da obnovi AOF datoteku u pozadini bez prekida usluge klijentima.

Prilikom pisanja na disk, podaci se prvo upisuju u tzv. bafer, a operativni sistem zatim u nekom trenutku uzima podatke i upisuje ih na disk. Pošto postoji vremenski razmak između te dve operacije, postoji i šansa da dođe do gubljenja podataka ukoliko u međuvremenu dođe do gašenja servera. Zato postoje opcije sinhronizacije koje podešavaju koliko često će se vršiti sinhronizacija bafera i diska. Postoje 3 opcije:

- *appendfsync always* - sinhronizacija se izvršava svaki put kada se nova naredba doda u AOF datoteku. Veoma sporo, veoma sigurno.
- *appendfsync everysec* - sinhronizacija se izvršava svake sekunde. Dovoljno brzo, a može se izgubiti samo 1 sekund podataka ukoliko dođe do neprilike.
- *appendfsync no* - bez podešavanja, operativni sistem vrši sinhronizaciju na svoj način. Brži i manje siguran metod. U uobičajenoj upotrebi, Linux bi ove podatke konfigurisao na svakih 30 sekundi, ali ovo zavisi od podesavanja samog kernela.

Optimalna opcija koja daje kompromis onoga što nudi i onoga što odnosi je opcija sinhronizovanja bafera i diska svake sekunde – ne opterećuje se disk, a u slučaju pada servera su izgubljeni podaci pristigli u toku jedne sekunde.

Primer korišćenja AOF datoteke

U konfiguracionom fajlu se doda linija kojom se omogućuje korišćenje AOF datoteke (slika 7.).

```
503 ############################## APPEND ONLY MODE ###############################
504
505 # By default Redis asynchronously dumps the dataset on disk. This mode is
506 # good enough in many applications, but an issue with the Redis process or
507 # a power outage may result into a few minutes of writes lost (depending on
508 # the configured save points).
509 #
510 # The Append Only File is an alternative persistence mode that provides
511 # much better durability. For instance using the default data fsync policy
512 # (see later in the config file) Redis can lose just one second of writes in a
513 # dramatic event like a server power outage, or a single write if something
514 # wrong with the Redis process itself happens, but the operating system is
515 # still running correctly.
516 #
517 # AOF and RDB persistence can be enabled at the same time without problems.
518 # If the AOF is enabled on startup Redis will load the AOF, that is the file
519 # with the better durability guarantees.
520 #
521 # Please check http://redis.io/topics/persistence for more information.
522
523 appendonly yes
524
525 # The name of the append only file (default: "appendonly.aof")
526 appendfilename "appendonly.aof"
```

Slika 7. *Konfiguracioni fajl koji dozvoljava korišćenje AOF datoteke*

```
C:\Program Files\Redis>redis-cli -p 6379
127.0.0.1:6379> set key1 value1
OK
127.0.0.1:6379> get key1
"value1"
127.0.0.1:6379>
```

Slika 8. *Konfiguracioni fajl koji dozvoljava korišćenje AOF datoteke*

Nakon upisa vrednosti *value1* pod ključem *key1*, automatski se kreira novi fajl pod nazivom *appendonly.aof*, koji nakon ove operacije upisa ima sledeći izgled:

appendonly.aof

```
1  *2
2  $6
3  SELECT
4  $1
5  0
6  *3
7  $3
8  set
9  $4
10 key1
11 $6
12 value1
13
```

Slika 9. *Izgled AOF datoteke nakon prve SET operacije*

Nakon nasilnog gašenja servera (na isti način kao i gore navedeni proces) i ponovnog pokretanja, izvršavaju se naredbe iz AOF datoteke. Zbog toga se pozivom naredbe GET za ključ *key1* uspešno pribavlja vrednost, koja je prethodnim gašenjem servera izgubljena.

Zaključak

Dakle, izrada rezervnih kopija podataka u Redisu i oporavak od neželjenih događaja su od krucijalnog značaja. Računari se kvare, instance na cloud-u nestaju i tako dalje: nemati backup baze znači ogroman rizik da podaci nepovratno mogu nestati.

Redis sistem je veoma jednostavan za pravljenje sigurnosnih kopija podataka jer se RDB datoteke mogu kopirati dok baza podataka radi. RDB datoteka se nikada ne modifikuje kada se jednom napravi, ali dok je u procesu izrade, koristi privremeni naziv, a kada se novi snimak baze potpuno završi, preimenuje se u svoju krajnju, konačnu destinaciju. Ovo znači da je kopiranje RDB datoteke potpuno sigurno dok se server pokreće.

Ukoliko se pokrene Redis server samo sa AOF metodom postojanosti podataka, takođe je moguće kopirati ovu datoteku kako bi se stvorile rezervne kopije. Datoteci će možda nedostajati završni deo, ali će Redis i dalje moći da je učita.

Oporavak od neželjenih događaja u kontekstu Redis-a je u osnovi ista priča kao i backup-ovi, plus mogućnost prenosa tih backup-ova u mnoge različite eksterne centre podataka. Na taj način se podaci osiguravaju čak i u slučaju katastrofalnih događaja u glavnom centru podataka gde se Redis pokreće i proizvodi svoje snimke baze.

Reference

1.https://redis.io/topics/persistence, Redis persistence

2.https://www.digitalocean.com/community/tutorials/how-to-back-up-and-restore-your-redis-data-on-ubuntu-14-04#conclusion, How To Back Up and Restore Your Redis Data

3.http://www.racunarstvo.matf.bg.ac.rs/MasterRadovi/2018_03_24_AnaSimijonovic/rad.pdf, Baza podataka Redis, Master rad