

*Računarstvo i informatika*

*Katedra za računarstvo  
Elektronski fakultet u Nišu*

# Napredne baze podataka **Key/Value Stores**

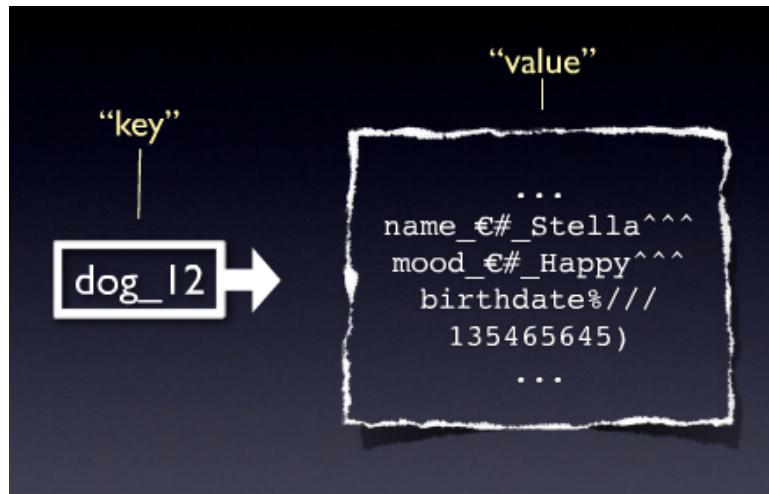
Zimski semestar 2020/2021

# Key/Value Stores

- Key/Value Databases
- Key/Value paradigma za pristup podacima obezbeđuje dobre performanse i dostupnost podataka.
- Jedna vrednost, jedan ključ, nema duplikata, izuzetno brzo.
- Skaliranje ogromnih količina podataka.
- Projektovane da podnesu velika opterećenja.
- Ključevi i vrednosti mogu biti kompleksni objekti.
- Konzistentnost podataka se može primeniti samo na operacije koje se odnose na jedan ključ.

# Key/Value stores

- Key/Value lookups (DHT)
- Klasična Hash tabela.
- Paradigma za skladištenje, pribavljanje i upravljanje podacima koji su smešteni u obliku asocijativnih nizova (associative array).
- Associative array = dictionary/hash.



# Key/Value Stores

- Dictionary predstavlja kolekciju objekata (objects/records) pri čemu svaki objekat može imati svoju kolekciju atributa i njihovih vrednosti.
- Podatak može biti predstavljen kao BLOB objekat pri čemu baza podataka ne razume strukturu podatka.
- Svaki objekat je jedinstveno identifikovan korišćenjem ključa i ključ se koristi za pronalaženja objekta u okviru baze podataka.

# Key/Value Stores

- Prednosti:
  - Efikasne pretrage podataka po ključu (predvidive performanse).
  - Mogućnost jednostavne distribucije podataka u cluster-u.
  - Fleksibilnost šeme – struktura podataka se može razlikovati od ključa do ključa.
  - Memorijska efikasnost – predstavljaju se samo atributi čije vrednosti postoje (kod relacionih baza podataka svaki podataka mora da ima sve attribute).
  - Ne postoji problem object-relational impedance mismatch.
  - Korišćenje relacione baze podataka u sprezi sa sistemom za keširanje forsira korišćenje key/value arhitekture.

# Key/Value Stores

- Nedostaci:
  - Nepostojanje kompleksnih upita (kompleksnih filtara nad podacima)
  - Nepostojanje stranih ključeva
  - Za kreiranje spojeva između različitih podataka odgovorna je aplikativna logika.
  - Nedostatak standardizacije



# Key/Value Stores

- Poznata rešenja:
  - Riak Basho
  - Redis
  - Memcached DB
  - Berkeley DB (Oracle)
  - Aerospike
  - LevelDB (Google)
  - DynamoDB (Amazon)
- Document oriented baze podataka predstavljaju specijalan slučaj Key/Value baza podataka.
- Neke od graph baza podataka interno koriste key/value paradigmu za skladištenje podataka uz dodatak relacija.

# Redis

- Redis predstavlja najpopularniju implementaciju key/value paradigmе (avgust 2015).
- **REmote DIctionary Server**
- Besplatan
- Open source (BSD licenca)
- Kompletno razvijen korišćenjem ANSI C jezika (portabilnost na različite platforme).
- Napredni Key/Value baza podataka koja se često opisuje kao Data structure server.
- Value može biti predstavljen korišćenjem različitih struktura podataka.

# Redis

- U osnovi Redis predstavlja in-memory dataset.
- Perzistencija podataka može da se ostvari na dva načina:
  - Snapshoting – kompletan skup podataka iz memorije se periodično snima na disk
  - Apend-only File (AOF) – nakon izmena podataka u memoriji, te izmene se upisuju u append-only datoteku na disku (journal)
- Podrazumevano sinhronizacija podataka između memorije i diska se vrši na dve sekunde. Po potrebi frekvencija može da se poveća ili smanji.
- U slučaju kompletног otkaza biće izgubljene sve izmene koje su se desile nakon poslednje sinhronizacije.

# Redis

- Zahvaljujući in-memory implementaciji Redis nudi odlične performanse u odnosu na RDBMS sisteme kod kojih svaka transakcija mora da se snimi na disk da bi bila potvrđena.
- In-memory implementacija dovodi do problem sa trajnošću podataka. Izmene nad podacima koji nisu persistenti ili nisu sinhronizovane mogu biti izgubljene.
- Ne postoji značajna razlika u performansama između operacija čitanja i pisanja.
- Redis server je implementiran kao jedan proces sa jednom niti.
- Ne postoji mogućnost paralelizacije operacija kod jedne instance Redis servera.

# Redis tipovi podataka

- Redis koristi *binary safe* ključeve.
- Za ključ se mogu iskoristiti bilo koje binarne sekvene, od jednostavnih stringova do sadržaja JPEG datoteka.
- Prazan string je validan ključ.
- Preporuke vezane za ključeve:
  - Maksimalna veličina ključa je 512MB
  - Treba izbegavati isuviše duge ključeve koji mogu da dovedu do pada performansi zbog operacija poređenja prilikom pretrage podataka.
  - Treba izbegavati isuviše kratke ključeve kako bi se povećala čitljivost podataka
  - Definisati šemu za imenovanje ključeva

# Redis tipovi podataka

- Redis podržava skladištenje podataka predstavljenih u vidu različitih struktura podataka:
  - Stringovi
  - Liste
  - Skupovi (Sets)
  - Sortirani skupovi (Sorted sets)
  - Bitmape (Bit arrays)
  - HyperLogLogs
- Prilikom rešavanja različitih problema veliku pažnju treba posvetiti izboru pravog tipa podataka.

# Redis tipovi podataka

- String je najjednostavniji tip podataka koje Redis podržava.
- Binary-safe
- Maksimalna veličina je 512MB.
- Tipične operacije za rad sa stringovima:
  - SET – definiše vrednost za zadati ključ
  - GET – pribavlja vrednost za specificirani ključ
  - INCR, INCRBY – atomična operacija za inkrementiranje vrednosti. Ne posotji mogućnost pojave race conditions.
  - 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



# Redis tipovi podataka

```
> set mykey somevalue          > set counter 100
OK                           OK
> get mykey                  > incr counter
"somevalue"                   (integer) 101
                               > incr counter
                               (integer) 102
                               > incrby counter 50
                               (integer) 152
```

# Redis tipovi podataka

- Redis lista je implementirana kao lančana lista stringova.
- Efikasne operacije dodavanja elemenata na početak i kraj liste
- Elementi liste su sortirani prema redosledu dodavanja u listu.
- Izuzetno pogodne kada treba dodavati nove elemente u izuzetno velike liste.
- Tipične 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 – izbacuje 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 elementa sa zadatim indeksom



# Redis tipovi podataka

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

```
> rpush mylist a b c  
(integer) 3  
> rpop mylist  
"c"  
> rpop mylist  
"b"  
> rpop mylist  
"a"
```

# Redis tipovi podataka

- Tipična primena za Redis liste:
  1. Kod društvenih mreža da se na jednostavan način dobijen poslednjih N poruka/slika/komentara
    - Twitter koristi Redis listu za čuvanje informacija o najnovijim porukama.
  2. Da se implementira komunikacija između dva procesa korišćenjem producer/consumer obrasca.

# Redis tipovi podataka

- Redis set predstavlja neuređenu kolekciju stringova.
- Elementi u skupu moraju biti jedinstveni odnosno nisu dozvoljeni duplikati.
- Tipične operacije:
  - SADD – dodavanje jednog ili većeg broja elemenata u skup
  - SISMEMBER – proverava se da li se zadata vrednost nalazi u skupu
  - SMEMBERS – pribavlja sve vrednosti iz skupa
  - SRLEM – 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



# Redis tipovi podataka

```
> sadd myset 1 2 3          > sadd news:1000:tags 1 2 5 77
(integer) 3                  (integer) 4

> smembers myset           > sadd tag:1:news 1000
1. 3                         (integer) 1
2. 1                         > sadd tag:2:news 1000
3. 2                         (integer) 1
                             > sadd tag:5:news 1000
                             (integer) 1
                             > sadd tag:77:news 1000
                             (integer) 1

> sismember myset 3         > smembers news:1000:tags
(integer) 1                   1. 5
> sismember myset 30        2. 1
(integer) 0                   3. 77
                             4. 2
```



# Redis tipovi podataka

- Redis hash predstavlja kolekciju atribut/vrednost parova.
- Svaki hash može da sadrži više od 4 milijarde parova.
- Pogodni su za predstavljanje objekata.
- Tipične operacije:
  - HSET/HGET – postavlja/probavlja string vrednost nekog od atributa unutar hash-a
  - HMSET/HMGET - postavlja/probavlja string vrednost većeg broj atributa unutar hash-a.
  - HLEN – vraća broj atributa koji su definisani u okviru hash strukture
  - HKEYS – vraća sve attribute koji postoje u hash strukturi
  - HGETALL – vraća sve atribut/vrednost parove koji posotje u okviru hash strukture



# Redis tipovi podataka

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

# Redis tipovi podataka

- Redis sortirani set predstavlja kolekciju jedinstvenih string vrednosti.
- Svakom elementu sortiranog skupa je dodeljena float vrednost – SCORE.
- Elementu u sortiranom skupu su uređeni prema vrednosti score-a na sledeći način:
  - Ako elementi A i B imaju različit score, A > B ukoliko važi da je A.score > B.score
  - Ako elementi A i B imaju isti score, na njih se primenjuje leksikografsko poređenje
- Tipične 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



# Redis tipovi podataka

```
> 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"
```

# Redis tipovi podataka

- Redis bitmap ne predstavlja poseban tip podataka već predstavlja skup bit orientisanih operacija koje mogu da se primenjuju na stringove.
- Pošto je maksimalna dužina string BLOB-a 512MB na raspolaganju je  $2^{32}$  bitova.
- Glavna prednost korišćenja je velika ušteda u prostoru naročito u situacijama kada je potrebno zapamtiti veliki broj true/false (yes/no, on/off) informacija.
- Tipične 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)



# Redis tipovi podataka

```
> setbit key 10 1          > setbit key 0 1
(integer) 1                (integer) 0
> getbit key 10           > setbit key 100 1
(integer) 1                (integer) 0
> getbit key 11           > bitcount key
(integer) 0                (integer) 2
```



# Redis tipovi podataka

- Operacije koje su nezavisne od konkretnog tipa podataka:
  - EXISTS – proverava da li specificirani ključ 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 vrednosti 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
```

# Redis tipovi podataka

- Redis omogućava definisanje ključeva sa ograničenim vekom trajanja
- Za svaki ključ je moguće definisati timeout nakon čijeg isteka će ključ i njemu pridružena vrednost biti obrisani.
- Timeout se može specificirati korišćenjem preciznosti na nivou sekundi ili mikrosekundi.
- Informacija o timeout-u se snima na disk ukoliko je Redis dataset perzistentan. Snima se trenutak u kome podatak prestaje da bude validan.
- EXPIRE – komanda kojom se definiše timeout
- PERSIST – komanda koja briše definisani timeout

# Redis transakcije

- U većini slučajeva Redis je single-threaded i sve operacije su:
  - Atomične
  - Konzistentne
  - Izolovane
- Trajnost (durability) je konfigurabilna i predstavlja kompromis između efikasnosti i sigurnosti podataka.
- Redis transakcija omogućava izvršavanje skupa komandi u vidu jedne celine.
- Redis transakcija obezbeđuje da se tokom izvršavanja komandi jedne transakcije ne mogu izvršavati komande izdate od strane nekog drugog klijenta.

# Redis transakcije

- Redis transakcija je atomična (procesiraju se sve komande ili nijedna).
- Komanda MULTI omogućava grupisanje/baferovanje komandi koje će činiti transakciju.
- Komanda EXEC izvršava komande grupisane u transakciju.
- Ukoliko dođe do greške prilikom baferovanja komandi Redis odbija izvršavanje transakcije.
- U slučaju da dođe do greške prilikom izvršavanje jedne od komandi iz transakcije ostale komande će se izvršiti.
- Ne postoji rollback mehanizam.

# Redis Publish/Subscribe

- Redis implementira jednostavnu Publish/Subscribe paradigmu.
- Publisher šalje poruke u kanal ne znajući da li postoje klijenti zainteresovani za prijem tih poruka.
- Subscriber prima podatke iz određenih kanala ne posedujući informaciju da li postoje klijenti koji šalju podatke u te kanale.
- Tipične komande:
  - PUBLISH
  - SUBSCRIBE
  - UNSUBSCRIBE

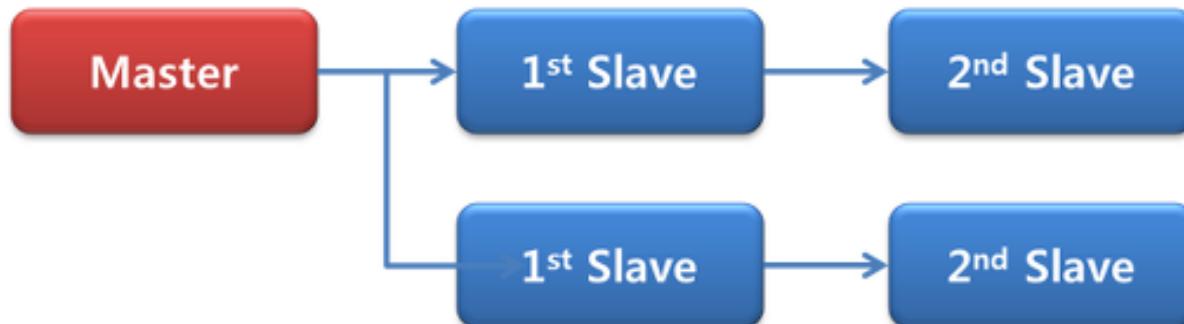
# Redis replikacija

- Redis podržava master/slave mehanizam replikacije.
- Podaci sa jednog Redis servera mogu se replikovati na veći broj slave servera.
- Slave server može postati master za neki drugi Redis server. Time se može implementirati stablo replikacije sa originalnim master serverom u korenu stabla (single-rooted replication tree).
- Redis slave server može da prihvata i write operacije čime se dozvoljava postojanje nekonzistentnosti između čvorova u stablu.
- Replikacije je pogodna u slučaju kada je potrebno obezbititi skalabilnost read operacija (ali ne i write operacija) ili redundantnost podataka.

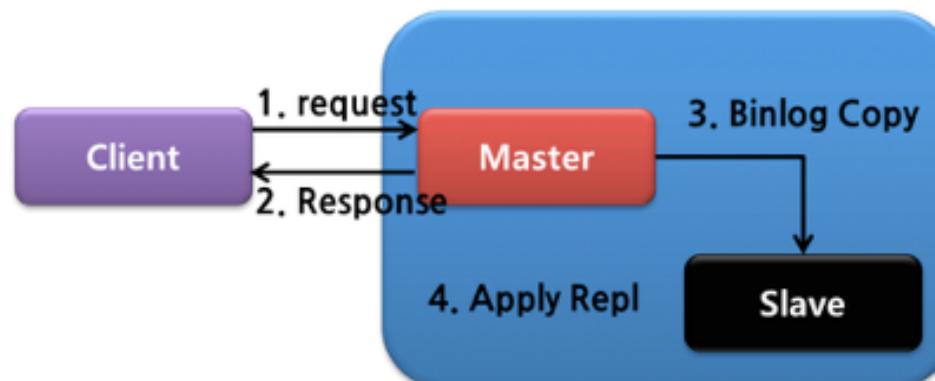
# Redis replikacija

Redis support an chained async replication.

```
slaveof <masterip> <masterport>
```

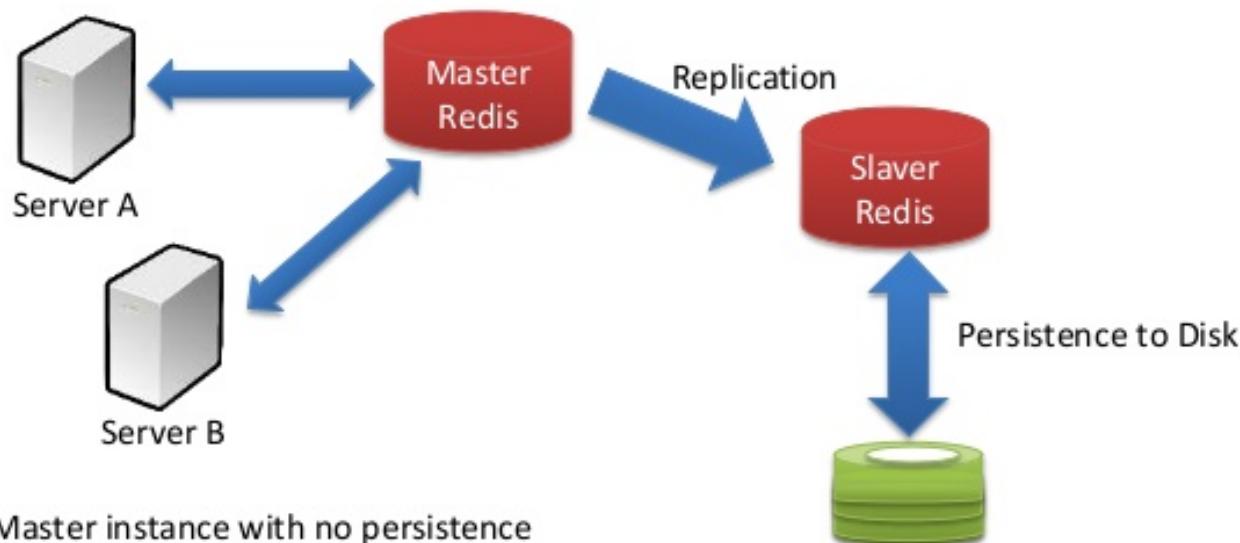


Async Model Flow



# Redis replikacija

## *Redis - Data Persistence*

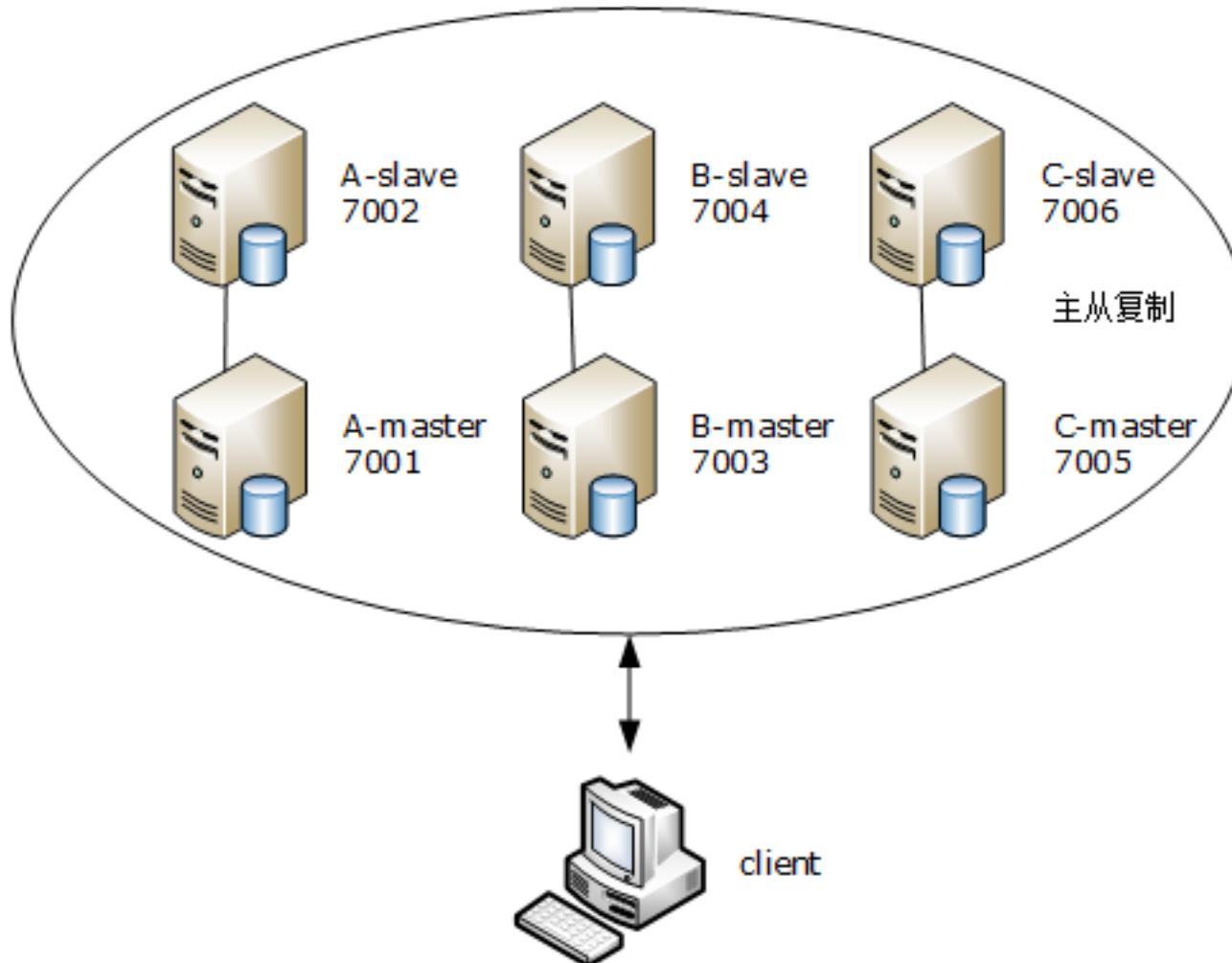


# Redis Cluster

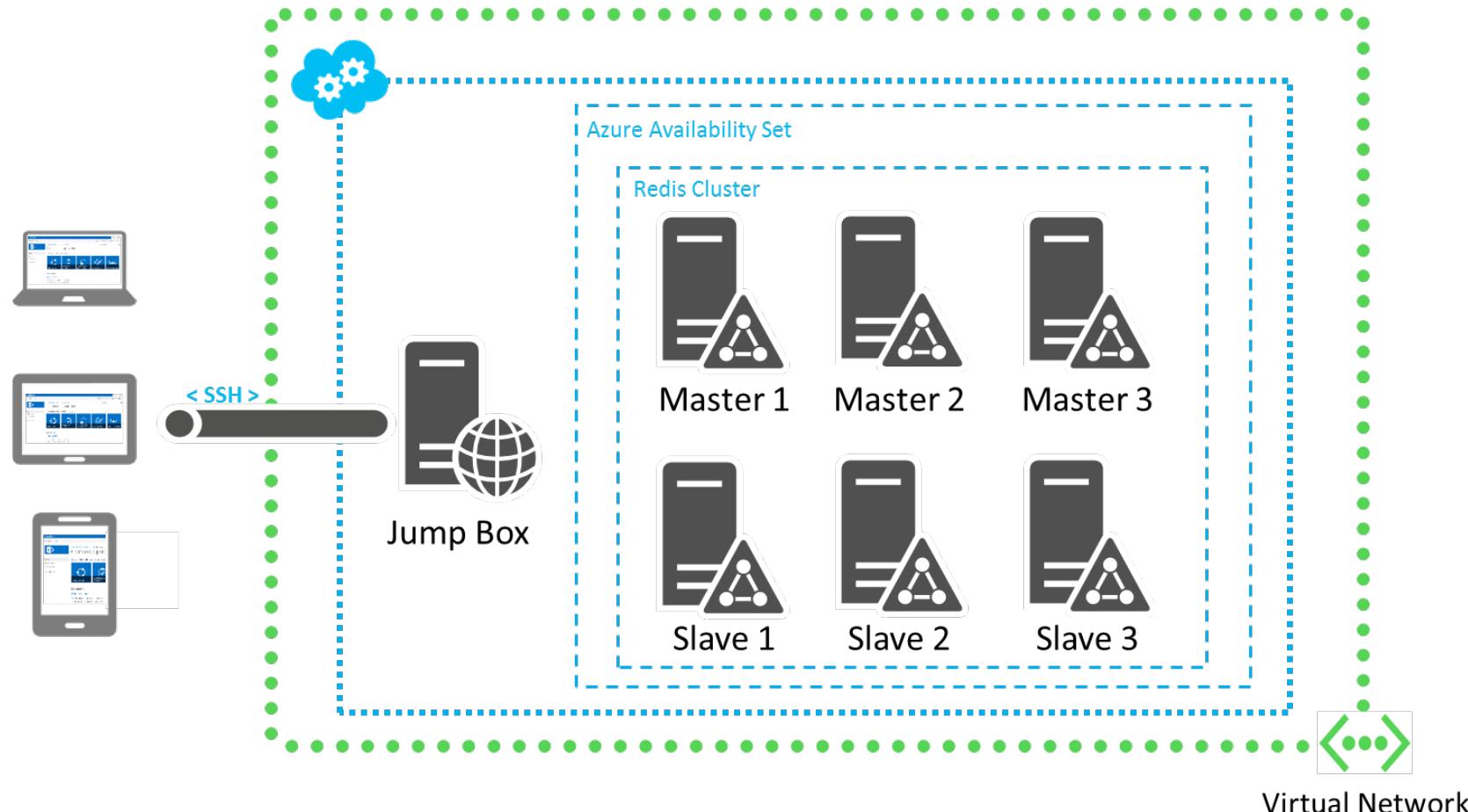
- Redis Cluster obezbeđuje automatsko particonisanje podataka između većeg broja instanci Redis servera.
- Redis Cluster obezbeđuje određeni nivo dostupnosti u situaciji kada neki od servera u klasteru otkaže. U slučaju većeg otkaza (kada otkaže veći broj master servera) klaster prestaje sa radom.



# Redis Cluster



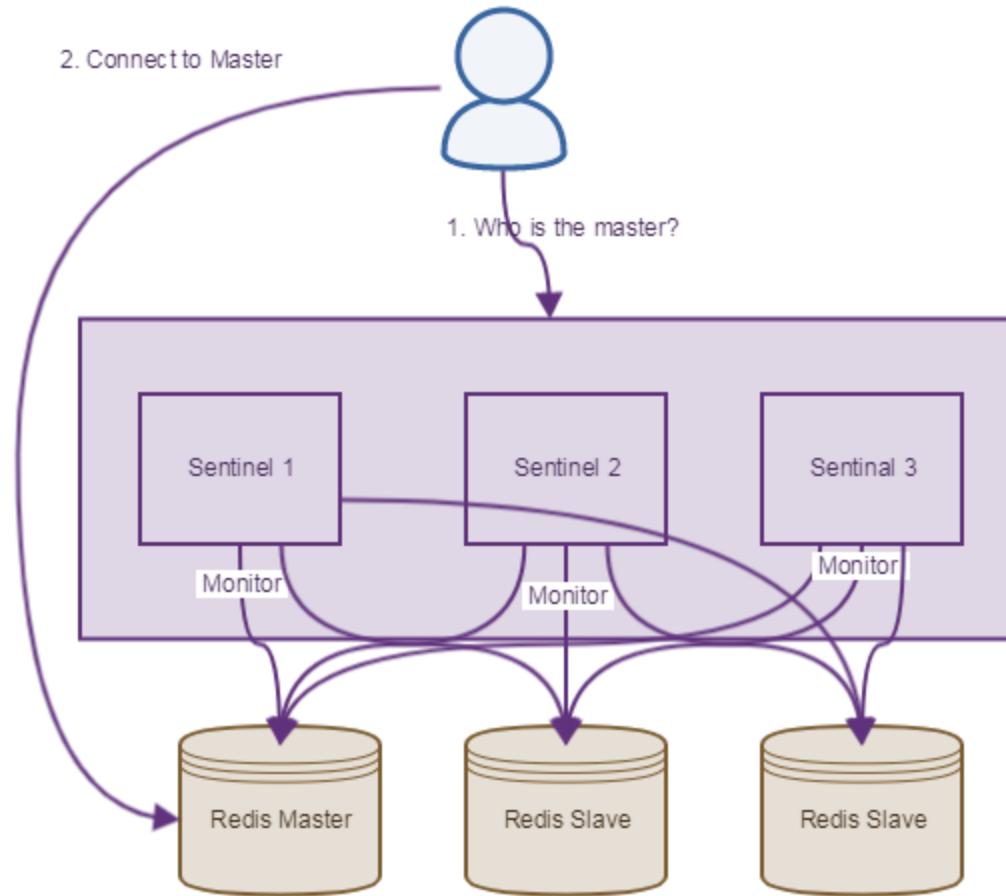
# Redis Cluster



# Redis Sentinel

- Redis Sentinel je distribuirani sistem namenjen za upravljanje Redis instancama:
  - Monitoring – proverava da li su Redis Master i Redis Slave instance dostupne
  - Notifikacija – obaveštava zainteresovane strane u slučaju da neka od Redis instanci postane nedostupna
  - Automatski failover – Ukoliko Redis Master čvor postane nefunkcionalan, Redis Sentinel započinje proces u kome se jedan od Redis Slave čvorova promoviše u novi Master čvor.

# Redis Sentinel





0

# Redis - Primena

twitter

Instagram

Pinterest



GitHub

craigslist

guardian.co.uk



digg™



DISQUS



bump technologies flickr® from YAHOO!

# Redis - primena

- Tipična primena:
  - Session cache
  - Full-page cache
  - Red poruka (message queues)
  - Brojači (counters)
  - Leaderboards, scoreboards
  - Publish/Subscribe
  - Real-time analytics

# Dynamo

- Razvijen od strane Amazona 2007. godine
- Distribuirani key-value store (DHT – distributed hash table)
- Jednostavan upitni model – za pristup podacima se koriste samo primarni ključevi
- Decentralizovan i simetričan sistem (P2P)
- Dodavanje/uklanjanje čvorova bez ikakve potrebe za ručnom rekonfiguracijom sistema
- Unapređena dostupnost (availability) a smanjena konzistentnost sistema (eventualy consistency)
- Smanjena latencija i povećana propusna moć
- Interno rešenja (nema mehanizama bezbednosti)

# Dynamo

- Namerno se žrtvuje konzistentnost podataka u korist dostupnosti (write operacija se uvek izvršava)
- Inkrementalna skalabilnost
- Simetričnost – svi čvorovi imaju identična zaduženja
- Decentralizacija
- Heterogenost

# Dynamo

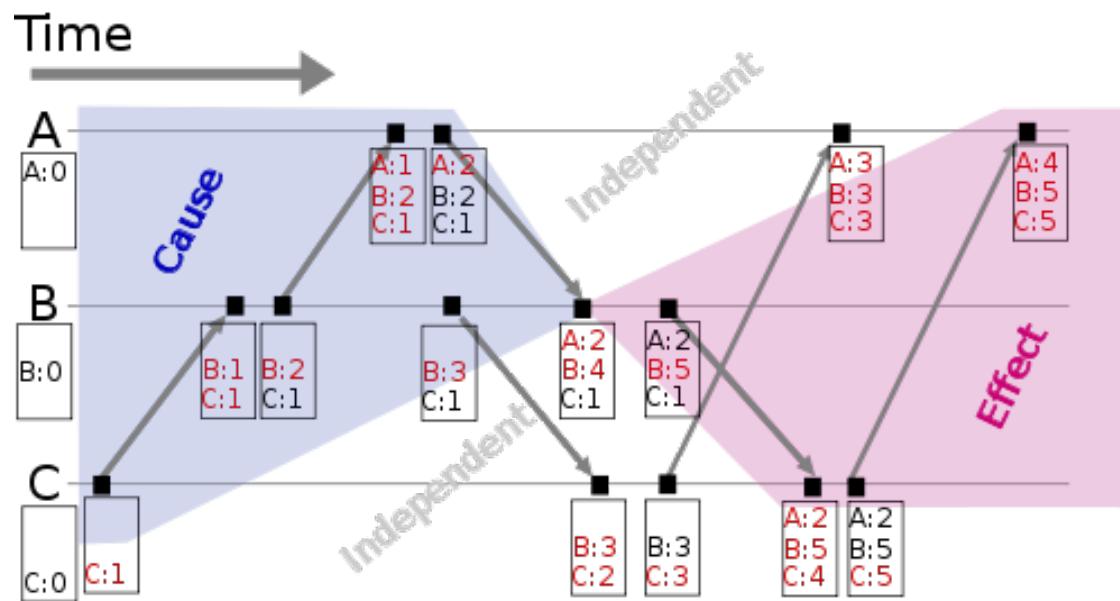
- Konzistentan hash store
  - Prsten čvorova (ring of nodes), svaki čvor dobija random poziciju prilikom uključivanja u sistem
  - Čvor koji se uključuje/napušta sistem utiče samo na prva dva suseda
  - Virtuelni čvorovi – jedan fizički čvor dobija nekoliko pozicija u prstenu
- Replikacija
  - N kopija – čvorovi sledbenici u prstenu
  - Scale-out

# Dynamo

- Čuvaju se različite verzije podataka
  - Svako ažuriranje obezbeđuje kontekst verzije objekta
  - Sva ažuriranja se ponavljaju u svim čvorovima u istom redosledu ali ne u isto vreme (eventual consistency)
  - Sistem je optimizovan za write operacije
  - Reševanje konflikata se vrši prilikom read operacije
  - Za rešavanje konflikata se koristi vector clock algoritam

# Dynamo

Vector clock algoritam za definisanje delimičnog uređenja događaja i detektovanje uzročno-posledičnih veza u distribuiranom sistemu



# Dynamo - operacije

- Read/Write zahtevi
  - Klijent šalje zahtev random čvoru koji ih prosleđuje odgovarajućem čvoru (čvor koji sadrži prvu repliku odgovarajuće particije – koordinator)
  - Koordinator prosleđuje R/W zahteve ka  $N$  prvih replika odgovarajuće particije
- Nivo konzistentnosti se podešava

Consistency level	Read / Write
ONE	1 replica
QUORUM	$N/2 + 1$
ALL	$N$

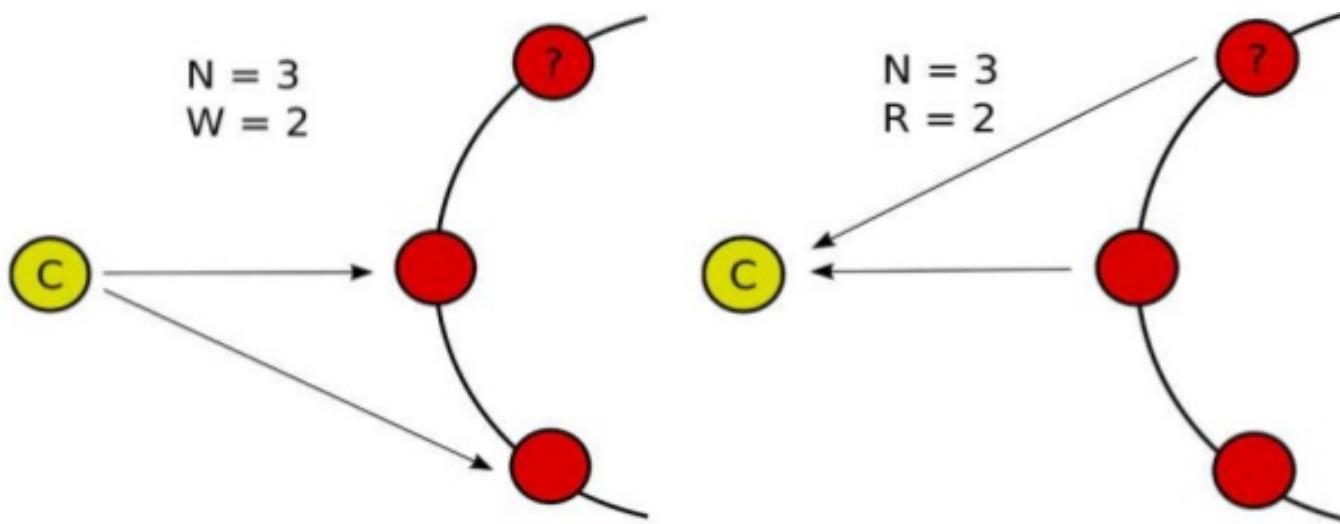


# Dynamo - operacije

- Konzistentnost

- Quorum protokol
  - R,W – minimalan broj čvorova koji moraju da učestvuju u R/W operaciji ( $R+W > N$ )
- Write
  - Koordinator generiše verziju i vektor clock
  - Koordinator šalje zahtev u N čvorova koji sadrže repliku podatka, ukoliko W čvorova potvrди operacija je uspešna
- Read
  - Koordinator šalje zahtev u N čvorova koji sadrže repliku traženog podatka, ukoliko R čvorova odgovori operacija je uspešna
  - Ukoliko postoje različite verzije podatka, razrešava se konflikt, generiše se važeća verzija podatka i ona se upisuje nazad (read repair)

# Dynamo operacije



$$R = N/2 + 1$$

$$W = N/2 + 1$$

$$R + W > N$$

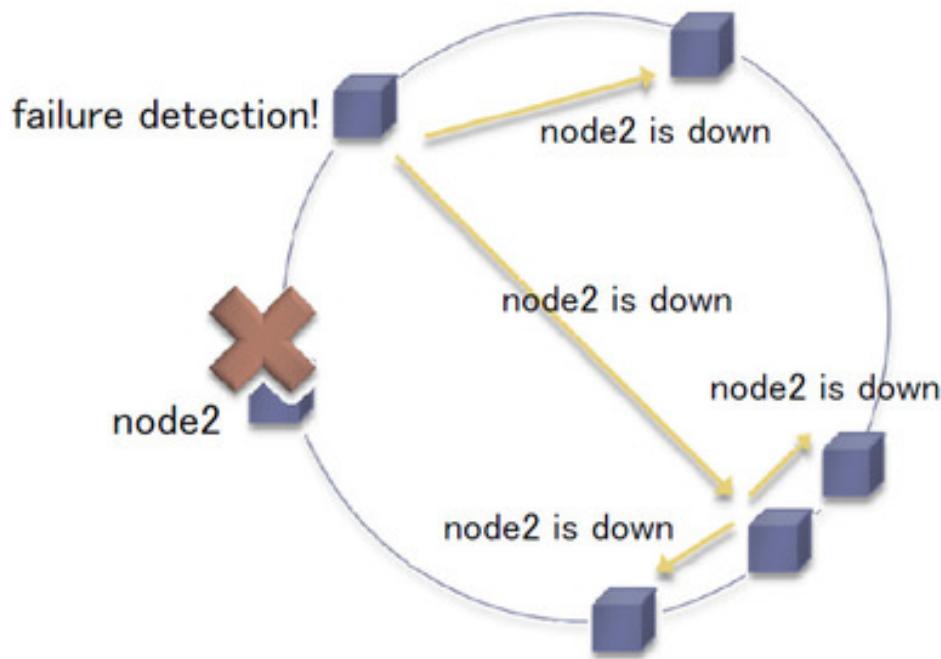


# Dynamo – obrada otkaza

- “Hinted handoff”
  - Dynamo uvek može da prihvati operaciju write
  - Ukoliko je čvor  $N_R$  koji sadrži repliku određenog podatka nedostupan, koordinator šalje podatke za upis novom čvoru  $N_{HH}$ , sa informacijom (hint) da je čvor  $N_R$  originalno odredište
  - Ovakve hinted write operacije nisu čitljive, odnosno read quorum može da izglosa da read operacija nije uspela
  - $N_{HH}$  čuva podatak sve dok  $N_R$  ne postane dostupan, šalje mu podatak, briše lokalnu kopiju

# Dynamo – obrada otkaza

- Gossip (epidemic) protocol
  - Po analogiji sa širenjem glasina na društvenim mrežama
  - Čvor prenosi informaciju drugim čvorovima koji se biraju po random principu



# Voldemort

- Baziran na Dynamo rešenju
- Distribuiran key-value store
- Razvijen od strane LinkedIn-a, sada open-source rešenje