- key-value storage

VALUES – LIST, SET, HASHES

- trzyma dane w memory – ram

- najszybsza baza, zapisuje więcej niz tylko stringi

- persistence, może byc ustawione by było przez dysk

- replication – scalable, master zmiany do inncyh

- ma język skryptowy LUA

- 5 typów

- no sql nie ma relacji, trzyma całą relację na raz – json

-- Customer {

Address : „”,

Orders: [O1, o2]

}

- nie ma nic narzuconego, od nas zalezy jak trzymamy

- RemoteDIctionaryService

- inne dokumentowe (jak mongo) trzymają dokument i index, redis inny

- tu po kluczu, dane w pamięci, a storage jako secondary consideration

- nie ma indexowania

- sql szukalo po query, mongo po index, a redis po kluczu specyficznym

- Dla win 64 unofficial version

- w wielu językach wsparcie bibliotek (Jedis, Jredis, JDBC redis, RUDS, Red protocol, lettuce dla java

- redis.io-> download -🡪 github , bin->release, pobieram zim 64, kilka exe kopiuję do redis dir,

Server uruchamia server a cli do testów

- klient z wystawionym portem default

- redis.io commends – wszystkie komendy są

- set – wklada wartość np. Set keyx : „Hello”, get keyx – zwróci nam wartość, del keyx – wyrzuca, to wszystko z klienta

- w następnym jak używać redis z 5cioma typami

2.

- publishing, subscribing , transactions

- comandline commands

- trudność to projektowanie jak skladowac i wydobywać dane z redis, zama bazka prosta

- strings, lists, sets, hashes – allow multiple fields contain multiple values , sortable sets – sorted by values

- string nie muszą byc zwykłym textem 120mb wielkość, mogą byc pliki, obrazy, obiekty

-- moze byc cały obiekt w serializowanej postaci

-- pozwala na low level bit operations

-- set, get, apped, incr decr – if number,

- getrange – substring

- mget mset – get set multiple values

- strlen – link to stored value

- key powinien dawać informacje o konkretnej entity, np. Set uset:1 „{’name’ : ‘joe’}” //value to json

- incr klucz zwieksza wartość jak to był numer

- append key „na koniec”

- getrange key:1 5 9 // oba włącznie

- mset key:1 “first” key:2 “sec”

- mget key:1 key:2

- strlen key:1 – (integer) 51

- list sorted insertion order, - linked list, references to naghbours

--fast to add first, last- rpush, lpush – wklada na początek,koniec

-- lset, lrem

-- lindex – by index, nie tak szybki, bo liked list

-- lrange – sublist

-- llen – link do listy

-- lpop, rpop

-- ltrim list to specified range, odcina koniec

- set for string, bez powtórek, szybkie operacje, w stałym czasie, jak elem raz dodany to potem ignoruje, nie trzeba sprawdzac , czy juz był

-- add, remove, contain – staly czas, nie zaleznie od ilosci operacji, union, intersaction,difference –bardzo szybko

-- sadd – dodaje elementy do setu, sadd post:1:likes „joe” „bob”

-- scard – ile elementow

-- sdiff, sinter (w obu),sunion – math set operation

-- sismember – contain

-- smembers – wypisze wszystkich

-- smove – elem from one set to other

-- srem – remove from set

- hashes – map string-string, optymalizacja, nie więcej niż 100 wierszy, więc szybkie, nie dużo miejsca, optymalizacja

-- HSET, HGET HSET USER:1:H NAME “JOE” ; HGET USER:1:H NAME

-- HMSET, HMGET – multiple – WYMIENIAMY POLA MAPY PO SPACJA

-- HGETALL

-- HDEL

-- HEXISTS

-- HINCRBY – INCREMENT STRING VALUE BY SPECIFIC

-- HKEYS, HVALUES – RETURN KEY/VALUES

-- HASH FOR PART OF STRUCTURE – MAIL OR EMAIL ADDRESS

- sorting sets sorted by score, fast adding, removing, updating, and getting by score

-- ZADD

-- ZCARD –ilosć elem w secie

-- ZCOUNT – liczy elementy w podamym przedziale score

-- zincrby – increment elem

-- zrenge – elements range by index

-- zrank - determine index of member in the set

-- ZREM – remove item

-- ZSCORE – score of given item

--zadd hs 120 “joe” 100 “bob”

-- zrange hs 0 4

-- zrange hs 0 4 WITHSCORES

-- JAK WRZUCE Z NOWYM SCORE, TO BĘDZIE UPDATE

-- to do trzymania jak dużo znajomych ktos ma, jak duz like itd.

- użycie jako message bus – wrzucanie na kolejkę

-- mam 2 klientów, pierwszy subscribe greetings, drugi publish greetings „hello” i u pierszego się pojawi wiadomość „hello”

-- subscribe greeting error – do 2, subscribe greet\* - do pattern się podłączy

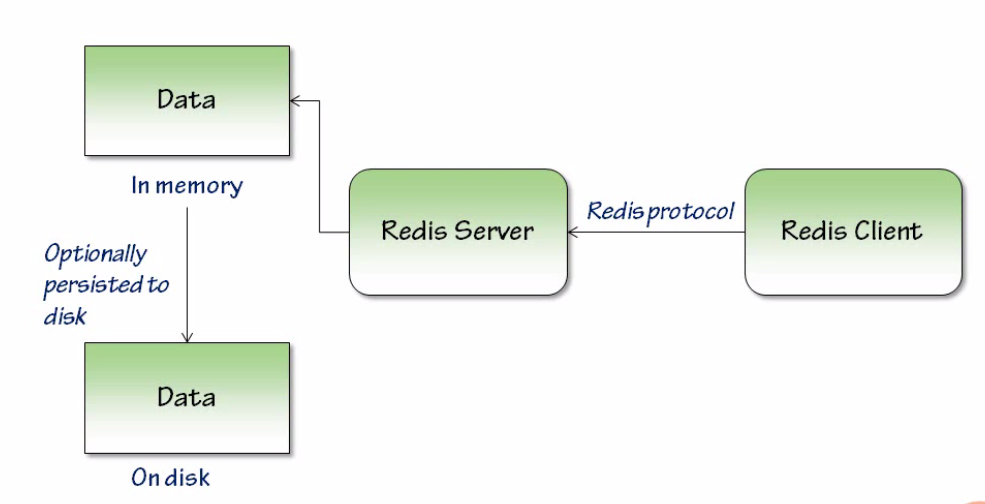
- Są transakcje w redis atomowe

-- nie ma rollbacj, komentdy są w kolejce do wykonania, dopiero exec zatwierdza, jak błąd pomiędzy to nic się nie stało

-- multi zaczyna transakcje, exec kończy

--Jak dam watch account przed multi, to jak robię exec, a cos się zmienilo na account, to nie wbije tylko zasygnalizuje - nil

3.



Szybki protoków przesyłu danych

- digital ocean – serwer linuxa, tu mozna wybrac jaka wersje i jaki rozmiar chcemy opcji, płatne

- apr –get update – updatuje paczke linux

- apt –get install build essential – daje gcc I inne narzedzia

- z /opt instalacja

- wget tar.gz – ścigam redis

- tar xvzf .. – rozpak

- clean, make – I zbudowane

- mozna si połczyc z klienta windows do linuxowego

-- redis-cli.exe –h port

- na prod kroki ze stronki

- redis.conf, command line, redis command (get I set na działajcym)– konfiguacja na 3 sposoby

-- w redis.conf dobrze udokumentowane zmienne, dobrze kopi przed zrobić

- 2 modele persistencji, mozna bez jako cash

-- rdb snapshot – szybka zrzutka, ale nie zawsze up to date, jak serwer pada moze być lekka zmiana danych

-- append only mode – wszystko kodowane i updatowane zanim kolejna komenda wykonana

-- id snapshot a potem aplikacja tego po kolei

-- definiujemy co ile sekund i ile minimum wierszy ma być update – mozna 3 opcje dla balansu

Jak duzo to częściej, jak mało to żadko

- mamy master instancje i kolejne jako slave – read only

-- redis-server –slaveof ip

- w seurity sekcji trzeba odblokowac requirepass i np. Insert mode, view mode

-- teraz po podłczeniu potrzebna autentykacja przez klienta – auth password

- jak na kliencie wywolam „monitor” – to wypisze wszystkie polecenia wszystkich klientów podłczonych do serwera

- ustawienie logów wolnych

-- na kliencie tym czasowa konfiguracja config set slowlog-log-slower-than 0 (0 loguje wszystko, ale mozemy tylko by jakies wolne logowało, można to pernament w redis.conf

-- potem komendy i jak wykonam slowlog get – wypisze, jak slowlog get 2 – 2 ostatnie – do sprawdzania performance

- Admin UI – user interface do redis

* Klient serviceStack.redis
* Tu opis interface I klasek w c#
* RedisClient (hisger level abstraction – getting data into string, - responsible get data into string), cachedClient(redis as cache), NativeClient(low level redis, full control, like using commandline, low level abstraction), ReditTyped (reialization/deser, generate unique id if not provided )
* W c# jest client.set client.get, jest cała lista omawianych poleceń w native
* - latwe transakcje , publish i subscribe

- trzeba myslec jak aplikacja używa danych

- jakie s powizania i jaka struktura powinna być

- któr czść danych bdziemy pobierać/zmieniać

- mamy wymagania i decydujemy co bedziemy musieli składować, w jakiej postaci – string, list sorted set

- np. User 3mamy jako string json mapowany na obiekt z polami

- lista ostatnich – jako json lista z user id usera, tam tylko 5 , wiec – usr:history:1 pobieram/tworzy, prepend – dodaje na poczatek, trim(0,4) – tylko 5 elem pobieram, łatwo identyfikować liste, bo ma nazwe z id usera na koncu

- kto nawicej czegoś, w sortedSet i przy modyfikacji tego pola musimy update set zrobić, tu identyfikacja to 1d usera – tylko 1, a score po którym sort to ilośćWszystkichSporzyć, czyli add to set jest usr:leadership, usr.Id, usr.total, jak klucz jest np. Nazwa, to trzeba przy jej zmianie updatować set