Ajuntament de Barcelona

Cibernàrium

Introducció a NoSQL

Capacitació Tecnològica per a Professionals i Empreses

Data

Barcelonactiva

Index

1. Redis

1.1 Introducció

Redis (REmote DIctionary Server): Es tracta d'un servidor de parelles clau-valor.

Open Source, Operatiu en memòria, Durabilitat opcional, Escrit en C, i un dels key-value stores més utilitzat en l'actualitat.

Hi ha bindings per molts llenguatges moderns.

1.2 Taula comparativa

Engine	MongoDB	Cassandra	Redis	HBase
Imatge				
Desenvolupador	10gen	Apache Software Fundation	Salvatore Sanfilippo	Apache Software Fundation
Data de llançament	2009	2008	2009	2010
Usuaris destacats	Foursquare, Craigslist	Twitter, Digg	StackOverflow	Facebook
Tipología	Document	Columna	Key-value	Columna

1.2 Taula comparativa

Llicència	AGPL	Apache	BSD	Apache
Llenguatge	C++	Java	C/C++	Java
Exemple d'ús	CMS, sistemes de vots, desat de comentàris	Banca, logging	Stock prices, Analítics, dades en temps real	Missatgeria en temps real
Millor utilitzar en...	Consultes dinàmiques, escriptures freqüents, poques consultes estadístiques	Grans volums d'escriptures en front a les lectures	Canvis freqüents de dades, escriptura freqüent, poques consultes estadístiques	Lectures aleatòries en grans bases de dades

1.2 Taula comparativa

Punts clau	Manté algunes propietats d'SQL, com els índexs i les consultes	Alta disponibilitat: es tracta d'un encreuament entre les idees de BigTable i Dynamo	Molt ràpid	Modelat sota la direcció de BigTable. Dissenyat per desar ingents quantitats de dades.
Control de concurrència	Locks	Multiversion concurrency control (MVCC)	Locks	Locks
URL	mongodb.org	cassandra.apache.org	redis.io	hbase.apache.org
Procés de replicació	Asíncron	Asíncron	Asíncron	Asíncron

1.2 Taula comparativa

Sistemes operatius	Linux, Mac, Windows	Linux, Mac, Windows	Linux, Mac, Windows	Linux, Mac, Windows
Desat de dades	Disc	Disc	RAM	Hadoop
Característiques	Consistència, tolerància a particions, persistència	Alta disponibilitat, tolerància a particions, persistència	Consistència, tolerància a particions, persistència	Consistència, tolerància a particions, persistència

1.3 Qui l'utilitza?

Branding:

- *Github*
- *Blizzard*
- *Stackoverflow*
- *Instagram*
- *Twitter*
- *Flickr*

En molts entorns, s'ha utilitzat com a reemplaçament de Memcache.

1.4 Redis vs Memcache

Pot pensar-se com una millora d'aquest sistema, amb les mateixes capacitats i més eines.

- Exposa funcionalitats existents, com l'autoexpiració de valors basats en marques de temps

- Alhora permet una persistència configurable en disc del contingut a memòria

- Exten el model per suportar estructures com a valors (i no només valors simples)

1.5 Característiques

Com a principals característiques, podem destacar:

- *Possibiltiat de persistència*
- *Escalabilitat horitzontal*
- *Suport per comptadors*
- *Estructures de dades complexes*
- *Operacions atòmiques*
- *Replicació master-slave*

Redis té un punt central de documentació:

http://redis.io/documentation

L'esquema és dels més flexibles dels que hem vist fins ara.

1.6 Model

Com veurem, l'estructura de dades en conjunt amb les operacions disponibles, ens permet utilitzar el software amb gran versatilitat

- Podem, per exemple, crear una cua de treball en memòria, només amb les eines exposades

El model, vist des de l'exterior, és simplement un mapa de claus apuntant als seus valors.

Una diferència clara amb altres repositòris clau-valor, és que els valors aquí no estan limitats a simples cadenes de caràcters. Pot contenir estructures del tipus:

- Llistes ordenades de cadenes
- Sets de cadenes (llistes sense ordre i sense repetits)
- Sorted sets de cadenes.

1.6 Model

Tipus de dades:

- ***Strings***: *El tipus més bàsic dins Redis. Està limitat a 512MB, i poden contindre qualsevol tipus de dades (incloent binaris). Pot ser utilitzat com a comptador, a través d'una reinterpretació de Redis com a enter. Permet operacions de codificació de dades.*

- ***Llistes***: *Son llistes d'strings, ordenades i sense control de duplicats. El nombre màxim d'elements son més de 4 bilions. Una capacitat a destacar és la de insertar / eliminar elements prop del cap i la cua de la llista en temps constant.*

1.6 Model

Tipus de dades:

- ***Sets****: Son col·leccions d'strings, sense ordre, i amb control de duplicats. Destacable en aquest constructe és que l'adició, eliminació i test d'existència es poden fer en temps constant. A més, permet accions entre sets que construeixen nous sets, com interseccions o unions.*

- ***Sorted sets (ZSET)****: La idea que segueixen és la mateixa que els sets, però afegeixen ordenació. El cost de les operacions és lleugerament superior.*

1.6 Model

Tipus de dades:

- ***Hash***: *És la representació de Redis sobre conjunts de clau-valor. És una bona aproximació als objectes dels llenguatges de programació.*

Dins els strings, podem guardar dades binàries, per el que el pròpi esquema s'exten a qualsevol dada que volem desar (per exemple, una imatge jpeg)

Podeu aprofundir més en els tipus de dades a la següent URI: http://redis.io/topics/data-types

1.7 Operacions

Les operacions disponibles asseguren l'atomicitat de la seva semàntica, tot i que redis no permet transaccions (conjunts d'operacions executades de forma atòmica). Podem considerar la següent llista:

- LPUSH: afegeix al cap de la llista un element
- RPUSH: afegeix a la cua de la llista un element
- LPOP: extreu l'element al cap de la llista
- RPOP: extreu l'element a la cua de la llista
- RPOPLPUSH: extreu l'element de la cua de la llista A i l'afegeix al cap de la llista B
- GETSET: obté el valor d'una clau, alhora que l'actualitza amb un nou valor

1.7 Operacions

- *MGET: obté els valors de diverses claus alhora*
- *MSET: desa els valors de diverses claus alhora*
- *SMOVE: mou un valor d'un set a un altre*

Podeu accedir a la llista sencera d'operacions a la següent URI
http://redis.io/commands

1.8 Com treballa

Cada valor te associades una serie d'operacions que poden no ser compartides amb altres tipus de valors. A continuació tenim un petit exemple d'operacions que s'admeten per el tipus de dades utilitzat:

String
- *get*
- *set*
- *incr*
- *decr*

List
- *ipush*
- *ipop*
- *llen*

Set
- *sadd*
- *srem*

1.8 Com treballa

El mètode de treball per defecte de Redis és totalment en memòria.

La persistència, configurable, es porta en dos vessants:

- Snapshotting: model de semi-persistència, on cada cert temps (o certes modificacions), el conjunt de dades és portat de memòria a disc asincronament.

- AOF: Append Only File, que és simplement un journal, on s'escriuen les operacions que modifiquen el conjunt de dades a mesura que es processen en l'estructura en memòria.

1.8 Com treballa

La replicació, igual que molts altres sistemes, porta associat un model molt similar a master-slave.

Aquesta replicació es porta en un esquema d'arbre

- Un master te n slaves
- Cada slave pot ser master d'altres slaves

Això pot portar a inconsistències entre instàncies (útil en sistemes basats en rols, on cada rol te accés a subsets diferents de dades, i només part d'aquestes dades poden ser comunes a altres nodes)

La replicació d'aquest sistema és molt útil per escalar en lectures (també ampliant la redundància), però pot arribar a penalitzar les escriptures.

1.8 Com treballa

Es pot utilitzar tant per motius d'escalabilitat com de redundància

Es slaves poden reconectar amb el master després d'un tall de comunicació automàticament

El rendiment de la plataforma depen totalment de l'ús que li donem

Si aprofitem les capacitats principals d'operabilitat en memòria per dades “no durables”, obtindrem un rendiment molt millor que el de les tradicionals RDBMS.

En qualsevol altre cas, la penalització del hardware de disc ens reduirà notablement el rendiment final de la plataforma.

1.9 Línia de comandes

Per treballar amb redis, utilitzarem un command line interface (redis-cli)

Podem utilitzar-lo com a comanda unitària

- redis-cli COMMAND ARGS

O com a interficie per interactuar amb redis de forma dinàmica

- redis-cli
- >> COMMAND1 ARGS
- >> COMMAND2 ARGS
- …

Per configurar redis, pot passar-se un fitxer amb les comandes de configuració, o interactivament en l'inici del servidor. Teniu disponible la informació a:
http://redis.io/topics/config

1.10 Utilitats

Redis incorpora una eina útil per consultar el rendiment de la plataforma, sobretot en l'esquema utilitzat.

L'eina s'anomena redis-benchmark

Training online
http://try.redis.io

Un recurs molt útil per tal de probar redis sense tenir-lo instal·lat

Ens permet probar les seves comandes, i exposa un tutorial pas a pas molt útil

1.11 Instal·lació

```
// Dependencies
aptitude install build-essential tcl
// Install
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make
make test
// Link
sudo make install
// Start server
redis-server
redis-server my_conf_file.conf
// Client line interface
redis-cli
redis-cli COMMNAD
redis-cli ping
```

1.12 Driver

Tenim diverses opcions per utilitzar aplicacions en conjunt amb Redis. Per homogeneitat, en aquest curs comptarem amb drivers (o interfícies) per Python.

redis-py
https://github.com/andymccurdy/redis-py

* Podeu trobar implementacions de drivers per Redis per molts llenguatges actuals

2. Exercicis sobre Redis

2.1 Implementació via shell
2.2 Implementació per aplicació
2.3 Benchmarking

2.1 Implementació via shell

En aquesta primera part pràctica, generarem una serie de projectes a partir de la consola del sistema que estem tractant.

- ***Twitter***: *generarem un clon del software conegut com Twitter. Com a definició, prendrem l'esquema d'usuari, tweet i seguidor.*
- ***Registre d'hotel***: *necessitem un sistema d'informació per el registre d'un hotel. S'ha de contindre informació sobre totes les habitacions, així com un historial tant d'usuaris com d'ús.*
- ***Log enriquit***: *volem tenir una peça de software que mantingui informació de log, concretament el còdi, una marca de temps, i un objecte, que contindrà informació arbitrària (fins i tot, altres objectes)*

2.1 Implementació via shell

- ***Item vault***: *un banc d'informació sobre objectes varis dins diverses col·leccions. L'objectiu és donar la versatilitat a un negòci d'objectes extranys per poder tenir un sistema d'informació amb totes les característiques d'aquests.*

- ***Enfermatim***: *base de dades d'informació sobre malalties. Aquest ha de contenir informació sobre geografía, causes, públic afectat, i estadístiques.*

- ***Workouts***: *tenim una gran cadena de gyms, i volem mantenir una base de coneixement sobre les màquines que tenim, així com dels clients i el seu creixement personal (muscular, dietes, salut, dies d'entrenament...)*

- ***Job box***: *es tracta d'un directòri d'ofertes de treball sobre tot el món. Volem informació sobre el treball, així com localització, empresa, perfils necessitats, i tota la informació de valor sota aquest concepte.*

2.2 Implementació per aplicació

En la segona part, generarem els mateixos projectes anteriors, a partir del llenguatge de programació que nosaltres escollim i els seus bindings sobre el sistema que tractem.

2.3 Benchmarking

Finalment, en aquesta tercera part, generarem scripts que omplirán les nostres bases de dades, i farem benchmarking sobre els processos d'escriptura i de lectura.

Barcelonactiva

Ajuntament de Barcelona

bcn.cat/barcelonactiva
bcn.cat/cibernarium