Elena Curti

DATABASE REDIS

Abstract

Con questo documento si vuole descrivere il database NoSQL key-value Redis. Viene inizialmente proposta una introduzione su Redis, spiegando perché convenga sceglierlo. In seguito, vengono descritte le principali caratteristiche di Redis: i tipi di dato, la performance, i meccanismi usati per la replica e la persistenza ed infine la sicurezza. Vengono poi date le istruzioni su come installare Redis e controllarne il corretto funzionamento. Nel capitolo finale vengono descritti i principali comandi base e viene poi mostrata un'applicazione d'esempio.

1. Introduzione

1.1 Redis

Redis (Remote Dictionary Server) è un archivio dati veloce, open source, in memoria e di tipo key-value. Lo sviluppatore iniziale di Redis è Salvatore Sanfilippo. Redis è scritto in C ed è usato come **database**, cache, broker di messaggi e coda. Redis offre tempi di risposta inferiori al millisecondo, consentendo milioni di richieste al secondo per applicazioni in tempo reale in settori come giochi e servizi finanziari. (Aws Redis)

Figura 1: Logo di Redis

1.2 Perché Redis?

1.2.1 Popolarità

Redis è stato votato database più amato su Stack Overflow Developer Survey per cinque anni di seguito, dal 2017 al 2021. (Redis)

1.2.2 Redis Client Library

È possibile connettere Redis ad un'applicazione usando una *Redis Client Library*. I linguaggi supportati sono: Node.js, .NET, Python, Java, PHP, C, Ruby e Drupal. (Develop with Redis clients)

1.2.3 Differenze dagli altri database

Redis ha reso popolare l'idea di un sistema che possa essere considerato **store e cache allo stesso tempo**. È stato progettato in modo che i dati vengano sempre modificati e letti dalla memoria principale del computer, ma anche archiviati su disco in un formato non adatto per il random data access. (Redis)

Inoltre, i comandi degli utenti, invece che descrivere delle query, descrivono specifiche operazioni da eseguire sui dati. Quindi i **dati devono essere memorizzati in un modo da permettere un recupero dati veloce**. Questo recupero non usa indici secondari, aggregazioni o altre caratteristiche comuni dei tradizionali RDBMS. (Redis)

A differenza di altri sistemi NoSQL che supportano solo le stringhe, **Redis supporta anche tanti tipi di dati astratti.** (Redis)

Mettendo a confronto Redis con altri database NoSQL key-value si ottengono i risultati riportati nel seguito.[1]

1.2.3.1 Data Loading

La seguente immagine mostra i tempi impiegati dai database per inserire 600.000 record.

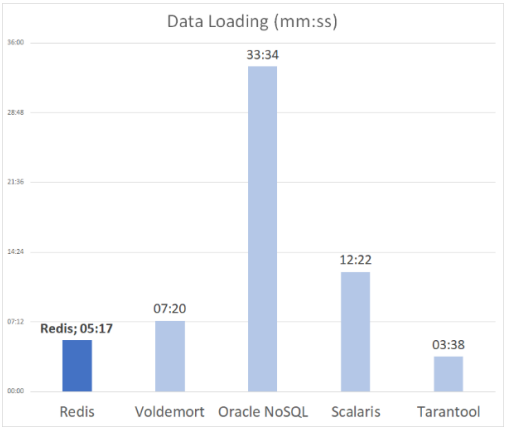

Figura 2: Data loading

In Redis, non si possono inserire 600.000 record alla volta, quindi il tempo riportato mostra la somma di due inserimenti da 300.000 record l'uno. In Redis ci sono voluti 2.9GB di RAM, mentre negli altri database 2 GB (Experimental Evaluation Of Nosql Databases).

1.2.3.2 Read e Update

La seguente immagine mostra i tempi impiegati dai database per eseguire 1000 operazioni di lettura e 1000 di aggiornamento nei 600.000 record.

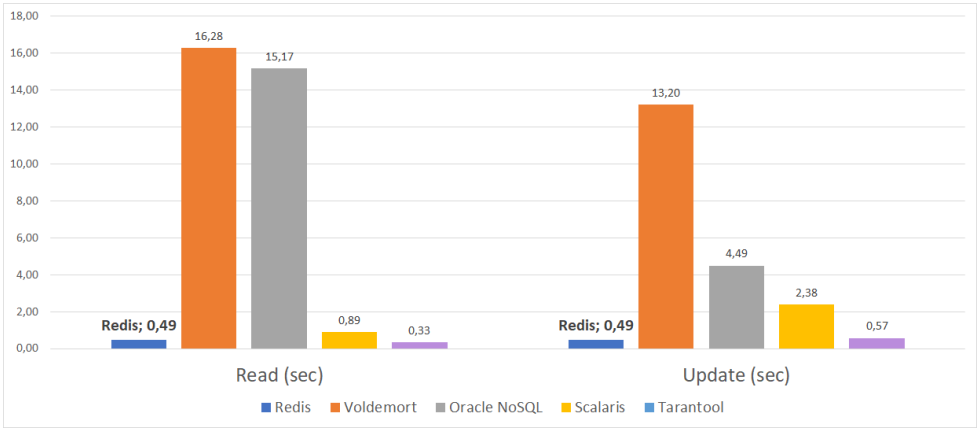

Figura 3: Read e Update

Redis ha avuto degli ottimi tempi durante i test eseguiti.

[1] I test riportati sono stati trovati nell'articolo *EXPERIMENTAL EVALUATION OF NOSQL DATABASES* di Veronika Abramova, Jorge Bernardino e Pedro Furtado (Polytechnic Institute of Coimbra e University of Coimbra), pubblicato nell'*International Journal of Database Management Systems (IJDMS)* Vol.6, No.3, nel giugno 2014.

2. Caratteristiche principali di Redis

2.1 Redis data types

Redis è un database key-value, cioè che mappa chiavi a tipi di valori. Come precedentemente accennato, Redis supporta molti tipi di dati astratti, oltre alle stringhe.

Alcuni tra i **principali tipi di dato** sono: (Redis data types)

- Strings. Tipi di dato più elementare, rappresentano una sequenza di byte.
- Lists. Liste di stringhe ordinate in base all'inserimento.
- Sets. Collezioni (senza ordine) di stringhe uniche (si comportano come i sets di Python). Il costo di aggiungere, rimuovere o testare l'esistenza di un elemento è di $O(1)$.
- Hashes. Collezione chiave-valore (si comportano come i dizionari in Python)
- Sorted Set. Collezioni di stringhe univoche, ordinate in base ad un punteggio associato ad ogni stringa.
- Bitmaps. Permette di eseguire operazioni bit a bit sulle stringhe.

Il tipo di un valore determina quali operazioni (o comandi) possono essere eseguite sul dato stesso. Redis supporta operazioni ad alto livello, atomiche e server-side. Integrando Redis con dei moduli, si possono aggiungere altri tipi di dato, come il **formato JSON**. (Redis)

Un valore non può superare 512 MB di dimensione.

2.2 Performance

Tutti i dati Redis risiedono in memoria. Gli archivi dati in memoria non richiedono ricerca sul disco, riducendo quindi i tempi di latenza ai millisecondi e permettendo di supportare una grande quantità di operazioni. (Aws Redis)

2.3 Replica e persistenza

Redis usa due meccanismi di persistenza: RDB e AOF.

2.3.1 RDB

RDB (Redis Database): consiste in uno snapshot che crea un salvataggio su disco dei dati immagazzinati nel server. RDB scatta in determinati momenti (impostati nel file di configurazione) mediante il comando

```
save <seconds> <changes> [<seconds> <changes> ...]
```

che realizza uno snapshot ogni `<seconds>` secondi, solo se sono state fatte almeno `<changes>` modifiche.

Il **rischio** principale di RDB è che l'interruzione improvvisa del server impedisca di salvare in maniera persistente le modifiche ai dati dopo l'ultimo snapshot. Per cercare di correggere questo problema, si predispongono più direttive `save` nel file di configurazione. (Redis: un DBMS NoSQL a dizionario)

La direttiva inserita di default (nel file */etc/redis/redis.conf*) è la seguente:

```
save 3600 1 300 100 60 10000
```

2.3.2 AOF

AOF (Append Only File): crea un file di log continuo in cui vengono registrate tutte le operazioni di modifica inviate al server. Questo file permette, al riavvio del server, di ripristinare in memoria l'ultimo dataset salvato su disco. Il sistema **crea file meno compatti** rispetto a RDB, ma il rischio di perdite di dati si riduce. AOF si può attivare da file di configurazione mediante il comando:

```
appendonly yes
```

Di default, questo meccanismo è disattivato. (Redis: un DBMS NoSQL a dizionario)

2.3.3 Uso dei due meccanismi

Per avere un'affidabilità comparabile a quella dei DBMS relazionali, bisogna abilitare entrambi i meccanismi, in modo che l'uno colmi le carenze dell'altro.

Se si è disposti a tollerare eventuali perdite di dati inseriti negli ultimi minuti, basta abilitare il solo RDB.

Invece, per avere un DBMS velocissimo che opera solo su RAM basterà disabilitare entrambe le opzioni. (Redis: un DBMS NoSQL a dizionario)

2.4 Sicurezza

Per analizzare la sicurezza di Redis, vengono riportati nel seguito alcuni dei risultati dello studio *A Comparison the Level of Security on Top 5 Open Source NoSQL Databases* di Preecha Noiumkar e Thawatchai Chomsiri (Mahasarakham University) pubblicato nel giugno 2014.

2.4.1 Crittografia dei dati e attacchi informatici

Redis **non supporta la crittografia dei dati**. Non viene eseguita alcuna crittografia dei dati nella comunicazione tra client e server perché Redis è stato progettato principalmente per funzionare rapidamente, quindi **non ha molti componenti per la sicurezza**. Per questo motivo, gli hacker con accesso alla rete dei server Redis saranno in grado di rilevare i dati mentre vengono comunicati. (A Comparison the Level of Security on Top 5 Open Source NoSQL Databases, 2014)

Redis comunque **non è vulnerabile ad attacchi Script Injection e Denial of Service**. (A Comparison the Level of Security on Top 5 Open Source NoSQL Databases, 2014)

2.4.2 Client-server authentication

In Redis **non è possibile effettuare l'autenticazione tra client e server**. (A Comparison the Level of Security on Top 5 Open Source NoSQL Databases, 2014)

2.4.3 Sniffing

Per impedire sniffing di dati (ad esempio durante la comunicazione tra client e server) può essere usato il protocollo SSL. I ricercatori dello studio hanno usato un tool chiamato **stunnel** e hanno scoperto che **si possono crittografare i dati in modo molto rapido e sicuro**, poiché stunnel utilizza il protocollo SSL con la compressione dei dati. (A Comparison the Level of Security on Top 5 Open Source NoSQL Databases)

Il client e il server sono stati collegati come illustrato nella seguente figura.

Figura 4: Collegamento stunnel tra client e server (A Comparison the Level of Security on Top 5 Open Source NoSQL Databases)

3. Installazione e uso iniziale

Per installare Redis seguire le istruzioni al link https://redis.io/docs/getting-started/.

Installare poi anche Redis Stack (https://redis.io/docs/stack/get-started/) per poter usare il tipo JSON.

Una volta installati Redis e Redis Stack, far partire il server. Aprire quindi il terminale e digitare:

```
redis-stack-server
```

Per connettersi al server, aprire un'altra finestra del terminale e digitare:

```
redis-cli PING
```

Se in risposta viene stampato `PONG` si è connessi correttamente al server.

3.1.1 Redis command line interface

Redis command line interface (`redis-cli`) è un programma utilizzato per inviare comandi e leggere risposte dal server Redis. Ha due modalità principali:

- Modalità REPL (Read Eval Print Loop) interattiva: l'utente digita i comandi Redis e riceve risposte.
- Esecuzione con argomenti aggiuntivi e con risposta stampata sullo standard output.

Di default, `redis-cli` si connette su localhost:6379. (Redis CLI)

4. Codice e applicazione di esempio

4.1 Comandi base delle chiavi

- `SET chiave valore` — Salva la coppia chiave-valore (`MSET` per inserire più coppie)
- `GET chiave` — Ritorna il valore associato alla chiave (MGET per ottenere i valori associati a più chiavi)
- `DEL chiave` — Elimina la chiave
- `EXISTS chiave` — Ritorna 1 se la chiave esiste; 0 altrimenti
- `SET variabile numero` `INCR variabile` — Crea una chiave con un certo valore, che viene poi incrementato
- `KEYS pattern` — Ritorna tutte le chiavi che fanno match con il pattern (`KEYS *` ritorna tutte le chiavi)

4.2 Comandi JSON

- `JSON.SET key path value [NX | XX]`
  Imposta il valore `value` (in formato JSON) nel `path` (percorso dell'albero JSON) della chiave `key`.
  Il path di default è la radice (`$`). Se il `path` esiste già, lo si aggiorna con il valore `value`.
  `NX`: Imposta la chiave solo se non esiste già
  `XX`: Imposta la chiave solo se esiste già

- `JSON.GET key [INDENT indent] [NEWLINE newline] [SPACE space] [paths [paths ...]]`
  Ritorna il valore al path specificato, in formato JSON.
  Il path di default è la radice (`$`). Possono essere inseriti più paths.
  `INDENT`: imposta la stringa di indentazione per i livelli innestati.
  `NEWLINE`: imposta la stringa da stampare alla fine di ogni linea
  `SPACE`: imposta la stringa che viene messa tra una chiave e un valore.

- `JSON.DEL key [path]`
  Cancella il valore con la chiave specificata. Il path di default è la radice (`$`). Path inesistenti vengono ignorati.
  `JSON.DEL` ritorna il numero di path eliminati (0 o più).

4.3 Applicazione di esempio

L'applicazione proposta mostra come effettuare inserimenti, stampe, modifiche ed eliminazioni con Redis.

Verranno memorizzati alcuni premi Nobel trovati su https://masterdataapi.nobelprize.org/2.1/nobelPrizes?offset=0&limit=664[2], la cui documentazione è visionabile al seguente link: https://app.swaggerhub.com/apis/NobelMedia/NobelMasterData/2.1.

NB: Prima di eseguire i comandi (e gli script) proposti, far partire il server e controllarne la corretta comunicazione con il client, come spiegato al capitolo 3. Installazione e uso iniziale.

4.3.1 Importare un documento JSON

Per importare in Redis i dati (contenuti nel file *nobelPrizes.json*), ho deciso di usare il pacchetto `redis` di Python (https://pypi.org/project/redis/). Ho usato le API di `redis` per collegarmi al database e inserirci i dati. Per fare ciò, ho scritto un piccolo script (*import_json.py*):

[2] Per semplicità, nell'applicazione sono stati salvati solo 22 premi Nobel. Inoltre, è stata modificata la struttura del file JSON originale. La versione usata nell'applicazione è inserita nel file *nobelPrizes.json*.

```python
import json
import redis
from redis.commands.json.path import Path

nome_file_json = "nobelPrizes.json"
nome_database = 'premi_nobel'

# Apro la connessione col server
r = redis.Redis(db=0)

# Recupero il contenuto del file...
with open(nome_file_json) as data_file:
    dati_json = json.load(data_file)
lista_json = list(dati_json)

# ... e lo salvo sul database
for i in range(len(lista_json)):
    r.json().set(nome_database+":"+str(i), Path.root_path(), lista_json[i])

```

Per eseguirlo: `python import_json.py`

Nello script vengono letti tutti i premi Nobel del file. Per l'i-esimo premio letto, viene salvata la chiave `premi_nobel:i` a cui è associato come valore il premio stesso in formato JSON.

Eseguire `redis-cli --raw` per entrare nella shell interattiva di Redis e poi verificare il corretto inserimento delle chiavi, digitando:

```
KEYS *
```

```
127.0.0.1:6379> KEYS *
premi_nobel:6
premi_nobel:10
premi_nobel:15
premi_nobel:3
premi_nobel:14
premi_nobel:5
premi_nobel:1
premi_nobel:17
premi_nobel:19
premi_nobel:2
premi_nobel:9
premi_nobel:18
premi_nobel:4
premi_nobel:20
premi_nobel:7
premi_nobel:21
premi_nobel:16
premi_nobel:0
premi_nobel:12
premi_nobel:11
premi_nobel:13
premi_nobel:8
```

4.3.2 Qualche stampa

Per stampare i dati del premio 1, si usa il comando:

```
JSON.GET premi_nobel:1
```

```
127.0.0.1:6379> JSON.GET premi_nobel:1
{"awardYear":1901,"category":"Literature","categoryFullName":"The Nobel Prize in
 Literature","prizeAmount":150782,"laureates":[{"id":"569","knownName":"Sully Pr
udhomme","portion":"1","sortOrder":"1","motivation":"in special recognition of h
is poetic composition, which gives evidence of lofty idealism, artistic perfecti
on and a rare combination of the qualities of both heart and intellect"}]}
127.0.0.1:6379>
```

Per ottenere dei dati più leggibili si può usare:

```
JSON.GET premi_nobel:1 INDENT "\t" NEWLINE "\n" SPACE " "
```

```
127.0.0.1:6379> JSON.GET premi_nobel:1 INDENT "\t" NEWLINE "\n" SPACE " "
{
        "awardYear": 1901,
        "category": "Literature",
        "categoryFullName": "The Nobel Prize in Literature",
        "prizeAmount": 150782,
        "laureates": [
                {
                        "id": "569",
                        "knownName": "Sully Prudhomme",
                        "portion": "1",
                        "sortOrder": "1",
                        "motivation": "in special recognition of his poetic composition,
llect"
                }
        ]
}
```

Per ottenere solo l'anno in cui è stato vinto il premio Nobel 1, si usa:

```
JSON.GET premi_nobel:1 awardYear
```

```
127.0.0.1:6379> JSON.GET premi_nobel:1 awardYear
1901
```

Per avere i nomi dei vincitori di un premio Nobel si usa:

```
JSON.GET premi_nobel:NUM_NOBEL $..knownName
```

(settando `NUM_NOBEL` con il valore desiderato)

```
127.0.0.1:6379> JSON.GET premi_nobel:1 $..knownName
["Sully Prudhomme"]
127.0.0.1:6379> JSON.GET premi_nobel:20 $..knownName
["Alain Aspect","John Clauser","Anton Zeilinger"]
127.0.0.1:6379>
```

4.3.3 Aggiornamento di un valore

Per cambiare il valore di una chiave si usa il comando:

```
JSON.SET premi_nobel:NUM_NOBEL $.chiave 'nuovo_valore'
```

(settando `NUM_NOBEL, chiave, nuovo_valore` con i valori desiderati)

```
127.0.0.1:6379> JSON.GET premi_nobel:1 prizeAmount
150782
127.0.0.1:6379> JSON.SET premi_nobel:1 prizeAmount '150783'
OK
127.0.0.1:6379> JSON.GET premi_nobel:1 prizeAmount
150783
127.0.0.1:6379>
```

4.3.4 Eliminazione di un valore

Per eliminare una chiave, si usa il comando:

`JSON.DEL premi_nobel:NUM_NOBEL $.chiave`

(settando `NUM_NOBEL` e `chiave` con i valori desiderati)

```
127.0.0.1:6379> JSON.GET premi_nobel:1 prizeAmount
150783
127.0.0.1:6379> JSON.DEL premi_nobel:1 $.prizeAmount
1
127.0.0.1:6379> JSON.GET premi_nobel:1 prizeAmount
ERR Path '$.prizeAmount' does not exist

127.0.0.1:6379> JSON.GET premi_nobel:1 INDENT "\t" NEWLINE "\n" SPACE " "
{
        "awardYear": 1901,
        "category": "Literature",
        "categoryFullName": "The Nobel Prize in Literature",
        "laureates": [
                {
                        "id": "569",
                        "knownName": "Sully Prudhomme",
                        "portion": "1",
                        "sortOrder": "1",
                        "motivation": "in special recognition of his poetic composi
llect"
                }
        ]
}
127.0.0.1:6379>
```

5. Conclusioni

Redis è quindi un database molto utile. Offre infatti tantissimi vantaggi, tra cui: prestazioni elevate, semplicità d'uso dei comandi, possibilità di memorizzare i dati in vari formati, integrazione rapida con la propria applicazione (mediante le Redis Client Library), possibilità di gestire le tecniche di persistenza e replica del proprio database.

6. Riferimenti Bibliografici

Abramova, V., Bernardino, J., & Furtado, P. (2014). *Experimental Evaluation Of Nosql Databases.* Tratto da International Journal of Database Management Systems (IJDMS): https://airccse.org/journal/ijdms/papers/6314ijdms01.pdf

Aws Redis. (s.d.). Tratto da Aws Amazon: https://aws.amazon.com/it/redis/

Develop with Redis clients. (s.d.). Tratto da Documentazione di Redis: https://docs.redis.com/latest/rs/references/client_references/

Noiumkar, P., & Chomsiri, T. (2014, Giugno). *A Comparison the Level of Security on Top 5 Open Source NoSQL Databases.* Tratto da ResearchGate: https://www.researchgate.net/publication/301633978

Redis. (s.d.). Tratto da Wikipedia: https://en.wikipedia.org/wiki/Redis

Redis CLI. (s.d.). Tratto da Documentazione di Redis: https://redis.io/docs/manual/cli/

Redis data types. (s.d.). Tratto da Documentazione di Redis: https://redis.io/docs/data-types/

Redis: un DBMS NoSQL a dizionario. (s.d.). Tratto da html.it: https://www.html.it/articoli/redis/