Redis, un moteur de base de données en mémoire

Redis (REmote Dictionary Server) est un moteur de base de données en mémoire développé à partir de 2009 par un développeur italien nommé Salvatore Sanfilippo. Ce dernier ainsi qu'un contributeur important de Redis (Pieter Noordhuis) sont recrutés par VMWare en 2010 pour se consacrer à plein temps sur le développement de Redis tout en laissant en licence libre BSD. Redis est écrit avec le langage de programmation C ANSI. Redis fait partie des solutions NoSQL et il se distingue par sa rapidité, son efficacité et sa légèreté. Par exemple, en terme de rapidité il est très difficile à battre, tant en lecture qu'en écriture. Il peut traiter plus de 100 000 opérations par seconde.

PRESENTATION

Redis est un moteur non relationnel qui opère principalement en mémoire. Il manipule des structures de données clé-valeur. Il est à la fois gestionnaire de ces types de données et un cache de données à la manière de memcached (un système d'usage général servant à gérer la mémoire cache distribuée. Il est souvent utilisé pour augmenter la vitesse de réponse des sites web (Facebook, Twitter, Wikipedia) créés à partir de bases de données. Il gère les données et les objets en RAM de façon à réduire le nombre de fois qu'une même donnée stockée dans un périphérique externe est lue.

TYPES DE DONNEES

Contrairement autres solutions NoSQL qui opèrent sur des documents (json, xml, etc), Redis manipule des paires clé-valeur, les valeurs peuvent être de cinq types de données : chaines, listes, ensembles, hachages et ensembles triés. Ci-dessous, nous détaillons ces cinq types de données.

Chaînes

Une chaîne en Redis est plus qu'une chaîne standard. Elle permet de stocker aussi de stocker des valeurs numériques et des valeurs binaires. Ces dernières permettent par exemple de stocker une image en mémoire. Ceci afin d'éviter le codage et décodage de format image. La taille maximale est de 512 Mo. D'autres commandes sont intégrées spécialement pour modifier des chaînes de type numérique INCR, INCRBY, DECR, DECRBY. Nous verrons toutes ces commandes dans les sections de travaux pratiques.

Listes

Les listes sont simplement des listes de chaînes.

Ensembles

Un ensemble est une collection non triée de chaînes. Quelques opérateurs ensemblistes sont disponibles au niveau de serveur Redis : union, intersection et différence.

Hachages

Il représente un dictionnaire, ce qui permet de stocker des objets par exemple (de la même façon qu'un document JSON). Le stockage est optimisé, ce qui fait que le hachage prend moins de place en mémoire.

Ensembles triés

L'ensemble trié se comporte comme un ensemble où chaque élément de l'ensemble est associé à un poids ou un score qui permet de trier l'ensemble.

PRATIQUES

Dans ces lignes, nous détaillons comment installer Redis, comment le configurer, donner quelques exemples de base pour manipuler les cinq types de données Redis, et enfin un projet complet Redis en utilisant CMake, Hiredis.

Installation

Redis est disponible sous forme de paquet dans plusieurs distributions : Debian, Ubuntu, etc. Sous Ubuntu ou Debian, le paquet est nommé : redis-server. Pour l'installer, il faut tout simplement lancer la commande :

```
#$> sudo aptitude install redis-server
```

Ensuite, une fois l'installation terminée, nous pouvons démarrer le serveur en lançant la commande :

```
#$> nohup redis-server
```

Avec cette commande le serveur démarre avec les paramètres par défaut. Si nous souhaitons les personnaliser alors, nous avons deux façons :

```
#$> nohup redis-server /etc/redis/redis.conf &
```

ou

```
#$> nohup redis-server --port 5555 --slaveof 127.0.0.1 8888 &
```

Dans la première commande les valeurs des paramètres sont renseignées dans le fichier de configuration redis.conf. Cependant dans la deuxième commande, les paramètres sont passés dans la ligne de commande.

Enfin, pour s'assurer que le serveur Redis est bien démarré, vous pouvez lancer la commande :

```
#$> Redis-cli ping
#$> PONG
```
Information serveur
Avant de personnaliser les paramètres de serveur Redis, il est utile de jeter un coup d'oeil sur la commande : INFO Pour lancer cette commande, il faut passer par redis-cli. Ce dernier est livré avec Redis-server. Redis-cli est une invite interactive qui permet d'envoyer des commandes au serveur.
```
#$> redis-cli
redis 127.0.0.1:6379> INFO
redis_version:2.4.14
redis_git_sha1:00000000
redis_git_dirty:0
arch_bits:64
multiplexing_api:epoll
gcc_version:4.6.2
process_id:5793
uptime_in_seconds:83
uptime_in_days:0
lru_clock:1410105
used_cpu_sys:0.00
used_cpu_user:0.04
used_cpu_sys_children:0.00
used_cpu_user_children:0.00
connected_clients:1
connected_slaves:0
client_longest_output_list:0
client_biggest_input_buf:0
blocked_clients:0
used_memory:726144
...
```
La commande INFO fournit un aperçu utile sur les informations et les paramètres de serveur Redis, y compris la version, ID de processus, la mémoire utilisée, la mémoire disponible, le numéro de port, l'adresse de serveur ; etc.
Configuration
Le serveur Redis possède plusieurs paramètres de configuration. Pour les personnaliser, nous pouvons lancer le serveur avec notre propre fichier de configuration. Afin de faciliter cette tâche, vous pouvez vous servir du fichier de configuration d'exemple redis.conf téléchargeable depuis cette adresse : https://github.com/antirez/redis. Voici un aperçu de fichier de configuration :
Utilisation de redis-cli
Dans cette section, nous utilisons redis-cli pour manipuler les types de données de base dans Redis.

```
redis 127.0.0.1:6379[1]> EXISTS mykey
(integer) 0
redis 127.0.0.1:6379[1]> APPEND mykey "hello"
(integer) 5
redis 127.0.0.1:6379[1]> APPEND mykey " world"
(integer) 11
redis 127.0.0.1:6379[1]> GET mykey
"hello world"
```
Je vérifie si mykey existe dans la base de données, puis je lui associée la valeur hello. Ensuite je concatène à cette valeur le mot word. En fin, avec la commande GET j'affiche le contenu de mykey. Vous remarquez que le serveur Redis répond toujours par le nombre de caractère du contenu de ma clé.
D'autres commandes peuvent être utiles si la valeur associée à une clé est de type numérique.

```
redis 127.0.0.1:6379> SET mykey "10"
OK
redis 127.0.0.1:6379> INCR mykey
(integer) 11
redis 127.0.0.1:6379> GET mykey
"11"
```

La liste des commandes concernant les chaînes se trouve à cette adresse : http://redis.io/commands#string
Avec Hache, nous pouvons stoker un objet. Supposons que nous avons besoin de stocker un

article. Avec Redis la commande sera :

```
redis 127.0.0.1:6379> hmset article:1 prenom "Pierre" nom "Durand" email
"pierre.durand@conceptit.fr" titre "Redis, quand on l'essaie, on l'adopte"
OK
redis 127.0.0.1:6379> hvals article:1
1) "Pierre"
2) "Durand"
3) "pierre.durand@conceptit.fr"
4) "Redis, quand on l'essaie, on l'adopte"
redis 127.0.0.1:6379> hget article:1 email
"pierre.durand@conceptit.fr"
```

D'autres commandes concernant cette fois les dictionnaires sont détaillées depuis cette adresse : http://redis.io/commands#hash
Si vous souhaitez utiliser les transactions, Redis met a votre disposition deux commandes MULTI et EXEC. MULTI commance une transaction. Toutes les instructions qui suivent seront stockées dans une file d'attente et ne seront exécutées qu'après l'appel de la commande EXEC. DISCARD permet d'annuler la transaction.

```
redis 127.0.0.1:6379> append mykey "hello"
(integer) 5
redis 127.0.0.1:6379> Multi
OK
redis 127.0.0.1:6379> append mykey " world"
QUEUED
redis 127.0.0.1:6379> get mykey
QUEUED
redis 127.0.0.1:6379> renamenx mykey macle
QUEUED
redis 127.0.0.1:6379> get macle
QUEUED
redis 127.0.0.1:6379> EXEC
1) (integer) 11
2) "hello world"
3) (integer) 1
4) "hello world"
```

APPLICATION CLIENTE
Dans cet article, j'ai choisi de vous donner un exemple d'application totalement développée avec le langage C++. Pour faire le pont entre notre application et Redis, nous utilisons Hiredis. Ce dernier existe en paquet pour debian et ubuntu sous le nom libhiredis-dev pour l'installer en même temps que cmake, il faut taper la commande :
sudo aptitude install cmake libhiredis-dev
Pour avoir cmake en interface graphique, il faut ajouter le paquet : cmake-curses-gui
Ingrédients
CMake,
Redis-server,
libhiredis-dev
Préparation
Dans un répertoire nommé par exemple "projet", créer deux sous répertoires : build, src.
Ensuite créer un fichier texte nommé CMakeLists.txt.
Dans le répertoire src, nous ajoutons notre programme main.cpp. Ce dernier contient les instructions nécessaires pour interroger notre serveur redis via bien sûr hiredis.
1
Dans le fichier CMakeLists.txt, ajouter ce contenu.

```
cmake_minimum_required(VERSION 2.8)
project(prjredis)
FIND_PACKAGE(HiRedis REQUIRED)
include_directories(${HiRedis_INCLUDE_DIR})
link_directories(${HiRedis_LIBRARY_DIR})
# Sources project
FILE(GLOB MY_SOURCES
src/main.cpp
)
SET (LINKS
${LIBHIREDIS_LIBRARIES}
)
add_executable(prjredis ${MY_SOURCES})
```

```
target_link_libraries(
prjredis
${LINKS}
)
```

Pour compiler notre projet lancer la commande depuis le repertoire build:

```
cmake ..
```

puis lancer l’exécutable prj.

```
./prjredis
```
