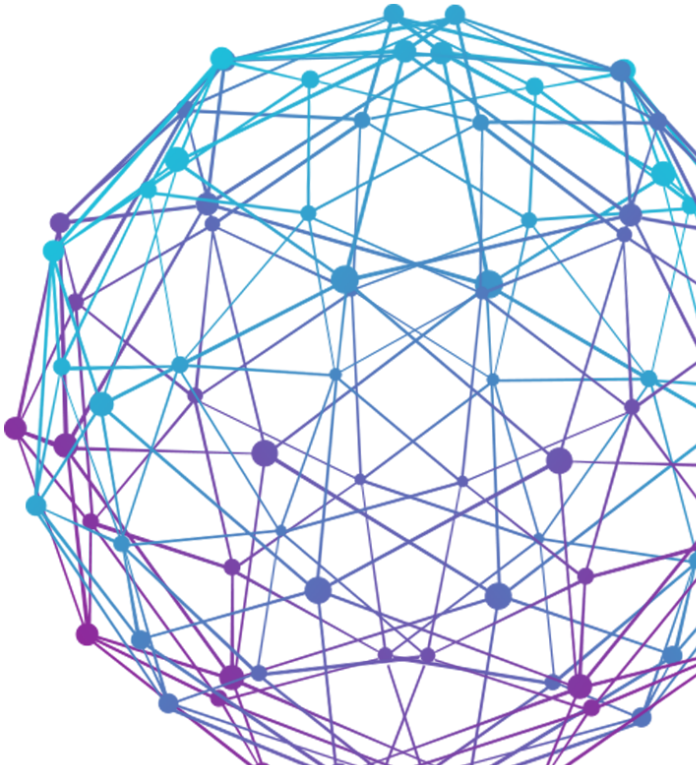
TP1 NoSQL

Compte rendu de travaux pratiques

Ryan YALA

INFOA3



23 janvier 2025

1 Introduction à Redis et aux Bases de Données NoSQL.

Tout, d'abord commençons par introduire les différents types de base de données NoSQL

2 Types de bases de données NoSQL

— **Clé-Valeur** : Les données sont stockées en paires clé-valeur, idéal pour des recherches rapides avec une clé unique (ex. : Redis, DynamoDB).
— **Colonnes** : Organisation des données par colonnes, adaptée aux analyses sur de gros volumes (ex. : Cassandra, HBase).
— **Documents** : Stockage sous forme de documents structurés comme JSON, parfait pour les applications web (ex. : MongoDB, Couchbase).
— **Graphes** : Représentation des données sous forme de graphes pour des relations complexes (ex. : Neo4j, ArangoDB).

3 Introduction à Redis

Redis, acronyme de **REmote DIctionary Server**, est une base de données NoSQL en mémoire, open source, offrant des performances ultra-rapides. Il a été conçu pour répondre aux limitations des bases de données traditionnelles dans des scénarios nécessitant des temps de réponse quasi instantanés et une gestion efficace des données complexes en temps réel.

Le besoin spécifique qui a conduit à l'introduction de Redis est lié à l'augmentation massive des volumes de données et à la nécessité de traiter ces données rapidement. Les bases de données relationnelles traditionnelles, bien qu'efficaces pour des transactions complexes, peinent à gérer des charges massives de lecture/écriture avec des temps de latence faibles. Redis, en stockant toutes les données en mémoire et en proposant des structures de données avancées comme les listes, les ensembles ou les hachages, répond parfaitement à des cas d'utilisation comme le caching, les systèmes de messagerie, les classements ou les tableaux de bord en temps réel.

4 Partie 1 : Démarrer un client et un serveur Redis depuis un terminal

4.1 1. Démarrer le serveur Redis

Avant d'utiliser Redis, il faut lancer son serveur. Voici les étapes :

1. Assurez-vous que Redis est installé sur votre système. Pour les distributions basées sur Debian/Ubuntu, utilisez :

code

```
sudo apt update
sudo apt install redis
```

2. Lancez le serveur Redis avec la commande suivante :

code

```
redis-server
```

Par défaut, Redis écoute sur le port **6379**. Une fois démarré, un journal d'activité s'affiche, indiquant que le serveur est opérationnel.

4.2 2. Démarrer un client Redis

Une fois le serveur Redis en cours d'exécution, vous pouvez interagir avec lui via le client Redis :

1. Ouvrez un nouveau terminal (sans fermer celui du serveur).
2. Lancez le client Redis avec la commande suivante :

code

```
redis-cli
```

3. Une invite s'affiche, indiquant que vous êtes connecté au serveur :

code

```
127.0.0.1:6379>
```

À partir de là, vous pouvez exécuter des commandes Redis, par exemple :

code

```
SET clé "valeur"
GET clé
```

4.3 3. Exécuter Redis en arrière-plan

Si vous souhaitez exécuter Redis en mode daemon (arrière-plan) :

1. Lancez le serveur Redis avec la commande suivante :

   code
   ```
   redis-server --daemonize yes
   ```

   Cela démarre le serveur Redis en arrière-plan.

2. Démarrez le client Redis comme expliqué précédemment :

   code
   ```
   redis-cli
   ```

3. Vérifiez si le serveur Redis fonctionne correctement avec la commande :

   code
   ```
   redis-cli ping
   ```

   Si tout fonctionne, la réponse sera :

   code
   ```
   PONG
   ```

4.4 4. Arrêter le serveur Redis

Pour arrêter le serveur Redis, vous pouvez utiliser :

```
redis-cli shutdown
```

Ou, si Redis fonctionne en arrière-plan, utilisez :

```
pkill redis-server
```

Redis est maintenant prêt à être utilisé pour vos projets. Si besoin, n'hésitez pas à explorer davantage ses fonctionnalités.

5 Partie 2 : Concepts et Commandes Redis

5.1 1. Gestion des clés simples (Key-Value Store)

Redis fonctionne principalement comme un magasin de paires clé-valeur, ce qui permet de stocker et récupérer des données rapidement.

— **Création d'une clé :**

code

```
SET demo "bonjour"
```

Cette commande stocke la valeur "bonjour" sous la clé `demo`.

— **Lecture d'une clé :**

code

```
GET demo
```

Résultat : la valeur associée à la clé `demo`.

— **Suppression d'une clé :**

code

```
DEL user:1234
```

Retourne `(integer) 1` si la clé est supprimée, `(integer) 0` si elle n'existe pas.

5.2 2. Incrémentation et décrémentation

Redis prend en charge les opérations arithmétiques simples.

— Création et incrémentation d'un compteur :

code

```
SET 1mars 0
INCR 1mars
```

Après chaque `INCR`, la valeur du compteur est incrémentée.

— Décrémentation :

code

```
DECR 1mars
```

Permet de diminuer la valeur du compteur.

5.3 3. Gestion des durées de vie des clés

Redis permet de définir une durée de vie pour une clé.

— **Vérification de la durée de vie d'une clé :**

code

```
TTL macle
```

Retourne `-1` si la clé est permanente.

— **Définir une expiration :**

code

```
EXPIRE macle 120
```

La clé expirera après 120 secondes.

5.4 4. Structures de données avancées

5.4.1 Listes

Les listes stockent des séquences ordonnées de valeurs.

— **Ajout d'éléments :**

code

```
RPUSH mesCours "BDA"
RPUSH mesCours "Service Web"
```

Ajoute les éléments à droite de la liste.

— **Lecture des éléments :**

code

```
LRANGE mesCours 0 -1
```

Retourne tous les éléments de la liste.

5.4.2 Ensembles (Sets)

Les ensembles stockent des valeurs uniques non ordonnées.

— **Ajout d'éléments uniques :**

code

```
SADD utilisateur "Augustin"
SADD utilisateur "Samir"
```

— **Lecture des éléments :**

code

```
SMEMBERS utilisateur
```

— **Union de deux ensembles :**

code

```
SUNION utilisateur autresUtilisateurs
```

Combine les éléments de deux ensembles.

6 Conclusion

Cette première partie permet d'explorer les fonctionnalités fondamentales de Redis, notamment la gestion des paires clé-valeur, les opérations atomiques d'incrémentation et de décrémentation, la définition de durées de vie pour les clés, ainsi que l'utilisation de structures de données avancées telles que les listes et les ensembles. Ces concepts illustrent la flexibilité et les performances élevées de Redis en tant que base de données NoSQL en mémoire. Pour approfondir vos connaissances et découvrir d'autres fonctionnalités avancées, il est recommandé de consulter la documentation officielle des commandes relatif à Redis : `https://redis.io/docs/latest/commands/`.

7 Gestion des ensembles triés et des hachages dans Redis

Dans cette section, nous explorons l'utilisation des ensembles triés (*sorted sets*) et des hachages (*hashes*) dans Redis, en nous basant sur les commandes exécutées lors du TP.

7.1 Ensembles triés

Les ensembles triés de Redis permettent de stocker des éléments associés à un score, facilitant ainsi le classement et la récupération ordonnée des données. Chaque élément est unique, et les scores déterminent leur ordre dans l'ensemble.

7.1.1 Ajout d'éléments avec `ZADD`

La commande `ZADD` ajoute des éléments à un ensemble trié avec un score spécifique :

Code

```
ZADD score4 19 "Augustin"
ZADD score4 18 "Ines"
ZADD score4 10 "Samir"
ZADD score4 8 "Philipe"
```

Ici, nous ajoutons quatre membres à l'ensemble trié `score4`, chacun avec un score distinct.

7.1.2 Récupération des éléments avec `ZRANGE`

La commande `ZRANGE` permet de récupérer une plage d'éléments dans un ensemble trié :

Code

```
ZRANGE score4 0 -1
```

Cette commande retourne tous les éléments de `score4` dans l'ordre croissant des scores :
— "Philipe"
— "Samir"
— "Ines"
— "Augustin"
Pour récupérer une plage spécifique, par exemple les deux premiers éléments :

Code

```
ZRANGE score4 0 1
```

7.1.3 Obtention du rang d'un élément avec `ZRANK`

La commande `ZRANK` retourne le rang d'un élément dans l'ensemble trié :

Code

```
ZRANK score4 "Augustin"
```

Cette commande indique que "Augustin" est au rang 3 (le comptage commence à 0).

7.2 Hachages

Les hachages dans Redis sont utilisés pour stocker des objets en tant que paires champ-valeur, ce qui est utile pour représenter des entités complexes comme des utilisateurs.

7.2.1 Ajout de champs avec `HSET`

La commande `HSET` ajoute des champs à un hachage :

Code

```
HSET user:11 username "youssef"
HSET user:11 age 31
HSET user:11 email "samir.youcef@polytech.fr"
```

Ici, nous créons un hachage `user:11` avec les champs `username`, `age` et `email`.

7.2.2 Récupération de tous les champs avec `HGETALL`

Pour obtenir tous les champs et valeurs d'un hachage :

Code

```
HGETALL user:11
```

Résultat :
— "username" : "youssef"
— "age" : "31"
— "email" : "samir.youcef@polytech.fr"

7.2.3 Ajout multiple de champs avec `HMSET`

La commande `HMSET` (désormais dépréciée au profit de `HSET` avec des paires champ-valeur multiples) permet d'ajouter plusieurs champs en une seule commande :

Code

```
HSET user:4 username "Augustin" age 5 email "august@gmail.com"
```

7.2.4 Incrémentation d'un champ numérique avec `HINCRBY`

Pour incrémenter la valeur numérique d'un champ :

Code

```
HINCRBY user:4 age 4
```

Après cette commande, le champ `age` de `user:4` passe de 5 à 9.

7.2.5 Récupération des valeurs avec `HVALS`

Pour obtenir uniquement les valeurs des champs d'un hachage :

Code

```
HVALS user:4
```

Résultat :
— "Augustin"
— "9"
— "august@gmail.com"

7.3 Conclusion

Cette deuxième partie à permis de manipuler des ensembles triés et des hachages dans Redis, illustrant ainsi la puissance et la flexibilité de ces structures de données pour gérer des informations complexes de manière efficace. Pour approfondir vos connaissances, consultez la documentation officielle de Redis : `https://redis.io/docs/latest/commands/`.

8 Partie 3 : Exemple d'utilisation du système Pub/Sub

8.1 Abonnement à un canal

Dans un terminal, utilisez la commande suivante pour s'abonner à un canal nommé `mesCours` :

Code

```
SUBSCRIBE mesCours
```

Le client reste alors en attente pour recevoir les messages publiés sur ce canal.

8.2 Publication d'un message

Depuis un autre terminal connecté au même serveur Redis, publiez un message sur le canal `mesCours` :

Code

```
PUBLISH mesCours "Nouveau cours sur MongoDB"
```

8.3 Résultat pour l'abonné

Le client abonné au canal `mesCours` reçoit instantanément le message :

Résultat

```
1) "message"
2) "mesCours"
3) "Nouveau cours sur MongoDB"
```

9 Abonnement à plusieurs canaux avec des motifs

Redis permet de s'abonner à plusieurs canaux en utilisant des motifs avec la commande `PSUBSCRIBE`.

9.1 Abonnement aux canaux commençant par un motif

Dans un terminal client, utilisez la commande suivante pour s'abonner à tous les canaux commençant par `mes` :

Code

```
PSUBSCRIBE mes*
```

9.2 Publication de messages sur différents canaux

Depuis un autre terminal :

— Publiez un message sur le canal `mesCours` :

Code

```
PUBLISH mesCours "Un autre cours est disponible"
```

— Publiez un message sur le canal `mesNotes` :

Code

```
PUBLISH mesNotes "Une nouvelle note est arrivée"
```

9.3 Résultat pour l'abonné

Le client abonné avec le motif `mes*` reçoit les messages des deux canaux :

Résultat

```
1) "pmessage"
2) "mes*"
3) "mesCours"
4) "Un autre cours est disponible"
1) "pmessage"
2) "mes*"
3) "mesNotes"
4) "Une nouvelle note est arrivée"
```

10 Gestion des bases de données dans Redis

Redis propose 16 bases de données par défaut, numérotées de 0 à 15. La base de données numéro 0 est utilisée par défaut.

10.1 Changer de base de données

Utilisez la commande `SELECT` pour passer d'une base de données à une autre. Par exemple, pour passer à la base numéro 1 :

Code

```
SELECT 1
```

10.2 Vérification des clés dans une base

Pour afficher toutes les clés d'une base de données :

Code

```
KEYS *
```

Si aucune clé n'est définie dans cette base, le résultat sera vide.

10.3 Revenir à la base initiale

Pour revenir à la base de données numéro 0 :

Code

```
SELECT 0
```

11 Remarque importante sur la persistance des données

Redis stocke les données en mémoire, et par défaut, elles ne sont pas immédiatement sauvegardées sur le disque. En cas de panne, les données non persistées peuvent être perdues. Pour éviter cela, il est recommandé de configurer des mécanismes de persistance comme :

— **Snapshots RDB** : Création périodique de clichés de la base de données pour une restauration rapide.
— **Fichiers d'appendice uniquement (AOF)** : Journalisation de chaque modification pour reconstruire l'état exact de la base.

Documentation officielle

Pour plus d'informations, consultez la documentation officielle de Redis :
— Pub/Sub : `https://redis.io/docs/latest/develop/interact/pubsub/`
— Gestion des bases : `https://redis.io/docs/latest/operate/oss_and_stack/`