Redis - Cheat sheet

Gwendoline Dössegger, Noémie Plancherel, Gaby Roch, Cassandre Wojciechowski

---

Redis vient de `Remote Dictionary Server`. C'est un système de gestion de bases de données NoSQL structuré selon un principe "clé-valeur". Ce software stocke les bases de données dans la RAM ce qui lui permet d'avoir de très hautes performances. Il est open source, multiplateforme et gratuit (il existe une version entreprise payante).

Database caching

Redis utilise la stratégie de mise en cache "look-aside".
Quant à la gestion du cache, elle dépend de la taille de mémoire maximale allouée au cache (`maxmemory`) ainsi que des politiques d'expulsion fournies. Ces dernières sont les actions effectuées pour déterminer quelle donnée sera supprimée afin d'en ajouter de nouvelles dès lors que la taille du cache atteint son maximum.

Pour cela il existe deux algorithmes:
- **LRU approximatif** : est une implémentation approximative de LRU (Least Recently Used). Cette solution permet d'offrir un meilleur coût mémoire. Elle consiste donc à déterminer selon un petit échantillon de clés celles ayant le temps d'accès le plus ancien pour la supprimer. Ainsi, plus la taille de l'échantillonnage de clés est grande, plus les performances seront améliorées.
- **LFU** (Least Frequently Used) : utilise un compteur pour définir la fréquence d'accès à l'objet. Lors de la création d'un objet, on lui assigne une valeur par défaut qu'on incrémente à chaque fois qu'on accède à la clé. Puis le compteur est décrémenté au fil du temps qui passe. Cette solution indique que plus la valeur du compteur est petite, plus la probabilité que la donnée soit expulsée du cache est grande.

Installation sous Debian

```
sudo apt install redis-server
```

Une installation manuelle est également possible: https://redis.io/download

Configuration

La configuration se trouve dans le fichier `/etc/redis/redis.conf`

On peut consulter toutes les configurations possibles sur https://redis.io/topics/config.

Exemple de configuration de départ:
- Créer un snapshot toutes les 5 minutes si au moins 10 entrées ont été modifiées

```
save 300 10
```

- Définir la place maximale autorisée pour la DB à 300MB

```
maxmemory 300mb
```

- Définir la politique d'expulsion du cache (lorsqu'on atteint `maxmemory`), voir *politiques d'expulsions* pour les différentes politiques

```
maxmemory-policy noeviction
```

Exécution Redis

Avec la commande `redis-cli` on obtient un terminal permettant d'interagir avec le serveur

Exemple de commandes

Une liste de quelques commandes basiques. Les commandes dépendent du type de données que l'on manipule. Voir toutes les autres commandes https://redis.io/commands

- Récupération d'une donnée (string) via sa clé

```
GET key
```

- Incrémentation/Décrémentation d'une valeur (integer)

```
INCR/DECR key
```

- Durée de vie d'une clé (TTL - time to live)

```
EXPIRE key seconds
```

- Tous les champs et les clé d'un hash

```
HGETALL key
```

- Récupération d'un élément d'une liste selon son index

```
LINDEX key index
```

- Ajout d'un élément au début d'une liste

```
LPUSH key element
```

Politiques d'expulsions

- `noeviction` : retourne une erreur si la taille du cache est atteinte
- `allkeys-lru` : applique LRU sur l'ensemble du cache.
- `volatile-lru` : applique LRU sur l'ensemble des clés qui ont un TTL.
- `allkeys-random` : efface aléatoirement une clé sur l'ensemble du cache.
- `volatile-random` : efface aléatoirement une clé sur l'ensemble des clés qui ont un TTL.
- `volatile-ttl` : efface les clés avec un petit TTL.