

Simulation d'algorithmes d'équilibrage de charge dans un environnement distribué

Kevin Barreau Guillaume Marques Corentin Salingue

Explication du sujet

Environnement distribué

- Base de données répartie sur plusieurs machines physiques
- Réplication multi-maîtres

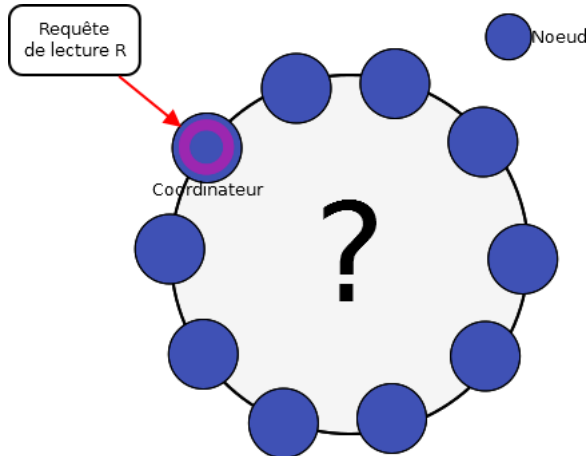
Algorithmes d'équilibrage de charge

- Créés par le client
- Basés sur la réplication des données

Simulation

- Comparaison de l'efficacité des différents algorithmes
- Objectif du projet \neq mise en production

Explication du sujet



Axes de développement

- Base de données (*Cassandra*)
 - Gestion des requêtes
 - Gestion de la réplication
- Application cliente (*Driver Java Cassandra*)
- Visualisation (*Graphite*)

Base de données Cassandra



Originellement créée et développée par **Facebook** en 2008 (maintenant un projet de la **Fondation Apache**), elle possède comme caractéristique d'être :

- NoSQL, orientée colonnes
- Open-source (licence Apache 2)
- Écrite en Java
- Décentralisée

Le choix de Cassandra



- Open-source
- Développement actif
- Proche du projet à réaliser
- Connaissances dans l'équipe

Solutions alternatives : HBase, CouchBase, CouchDB, from scratch...

Fonctionnement de Cassandra

Gestion des requêtes : affectation

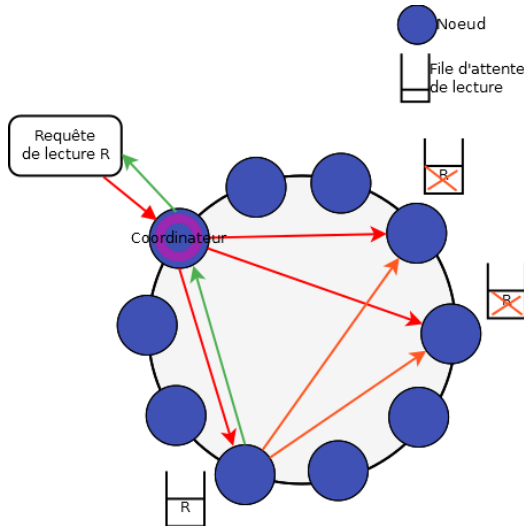
De base

- Requêtes de lecture pour certains noeuds
- Renvoie donnée entière pour une, *digest* pour les autres
- Suppression impossible

Modifié

- ✓ Requêtes de lecture pour tous les noeuds
- ✓ Renvoie donnée entière pour tous
- ✓ Suppression possible

Fonctionnement de Cassandra



Fonctionnement de Cassandra

Gestion des requêtes : réaffectation

De base

- Système inexistant

Modifié

- ✓ Compteur de requêtes assignées
- ✗ Algorithmes d'assignation
- ✗ Assignation

Fonctionnement de Cassandra

Gestion de la réplication

De base

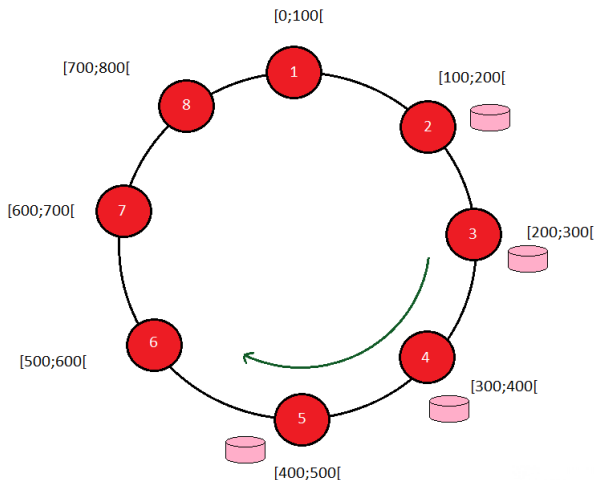
- Placement des copies d'un objet sur les noeuds suivant dans l'ordre du cercle

Modifié

- ✓ Placement des copies suivant différentes fonctions de hachages

Fonctionnement de Cassandra

Stratégie de réplication de base



Fonctionnement de Cassandra

Gestion de la popularité ✖

Paramètres

r = Nombre de requêtes total effectuées durant l'intervalle de temps T ;

n = Nombre de noeuds dans le réseau ;

p = Popularité d'un objet ;

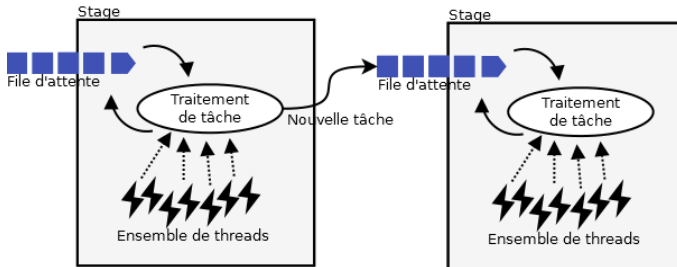
k = Nombre de copies de l'objet.

- Augmenter le nombre de copies si $2 \times \frac{r}{n} \geq \frac{p}{k}$ vraie.
- Diminuer le nombre de copies si $\frac{r}{2n} \leq \frac{p}{k}$ vraie.

Architecture de Cassandra

Staged event-driven architecture (SEDA)

- **Stage** → emplacement pour réaliser des tâches
 - **File d'attente** → messages de tâches à traiter
 - **Threads** → exécuteurs de tâches



Architecture de Cassandra

Staged event-driven architecture (SEDA)

- **Stage** → emplacement pour réaliser des tâches
 - **File d'attente** → messages de tâches à traiter
 - **Threads** → exécuteurs de tâches

Stages présents dans Cassandra :

- READ
- **READ_REMOVE**
- MUTATION
- GOSSIP
- ...

Point technique : Réplication

Solution initiale		Solution implémentée	
Donnée n° 1	Donnée n° 2	Donnée n° 1	Donnée n° 2
$H_0(c1)$	$H_0(c2)$	$H_0(c1)$	$H_0(c2)$
1er réplica	1er réplica	1er réplica	1er réplica
$H_1(c1)$	$H_1(c2)$	$H_1(H_0(c1))$	$H_1(H_0(c2))$
2nd réplica	2nd réplica	2nd réplica	2nd réplica
$H_2(c1)$	$H_2(c2)$	$H_2(H_0(c1))$	$H_2(H_0(c2))$

Application cliente

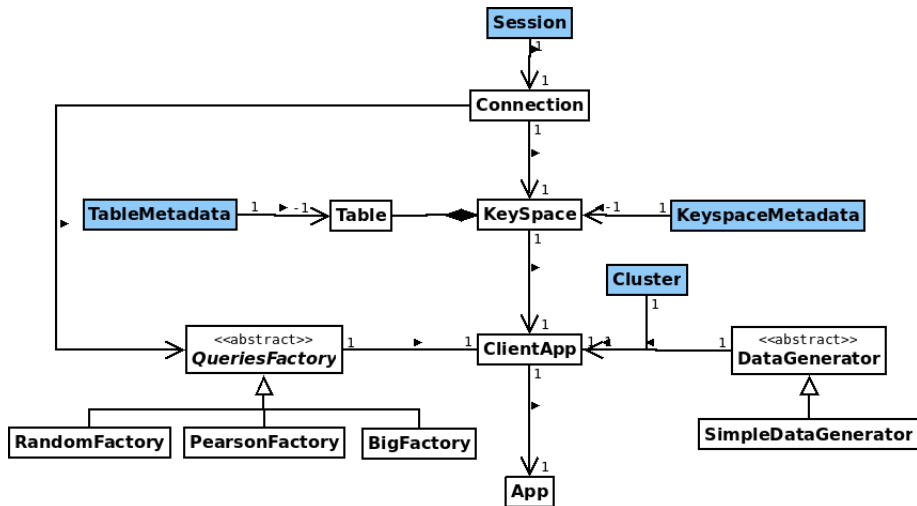
Technologies employées

- Développé en Java
- Utilisation d'un pilote informatique

Pilote utilisé

- DataStax Java Driver 2.0
- Développé par l'entreprise DataStax
- Communication avec la base de données Cassandra

Architecture du client



Fonctionnement du client

Initialisation

- Connexion à la base de données
- Choix du keyspace

Console

L'utilisateur saisie la commande qu'il souhaite exécuter, notamment :

- Changement de cluster
- Création de jeu de données
- Exécution d'un générateur de requêtes

Fonctionnement du client

```
[guillaume@t client]$ java -jar target/client-0.9.jar
Simulassandra Client
Please, enter the host address :127.0.0.1
Checking if 127.0.0.1 host is reachable.
Connected to clustersimul
Datacenter: datacenter1; Host: /127.0.0.1; Rack: rack1
Datacenter: datacenter1; Host: /127.0.0.3; Rack: rack1
Datacenter: datacenter1; Host: /127.0.0.2; Rack: rack1
Datacenter: datacenter1; Host: /127.0.0.5; Rack: rack1
Datacenter: datacenter1; Host: /127.0.0.4; Rack: rack1
Please, enter the keyspace name :test
You are now using keyspace test
```

```
> help
```

```
Lists of commands available :
```

```
- help : show this list
- import <file> : execute cql queries written in the file
- switchks <ks> : switch to keyspace ks
- queries <qf> <s> <ns> <nq> : execute queries generated by the seed s on the current keyspace
with the queries factory <qf>. <ns> is the number of simulation, <nq> the number of queries in each simulation.
- showksdata : show current keyspace metadata
- lstable : list tables available in the current keyspace
- showtabledata <t> : show table t metadata
- createdatafile <file> <nb_tables> <nb_rows> <data_length>
Create or write in file <file> CQL queries to create <nb_tables> with <nb_rows>.
- quit : quitter le programme
```

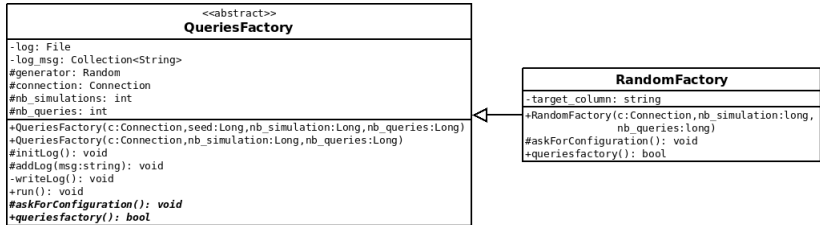
```
> queries RandomFactory 333 5 10
```

```
Target column : key
```

```
Starting quering.
```

```
End (166ms).
```

Générateur de requêtes



Personnalisable

- Possibilité d'ajouter des générateurs de requêtes
- Choix du générateur

queries NomGénérateur <seed> <nb_simulations>
<nb_requetes>

Tests

Tests

Environnement

- Les tests de mesures de performances se déroulent dans un réseau d'Amazon EC2 de 10 noeuds.
- La base de données est composée de 10 000 objets de taille unique.

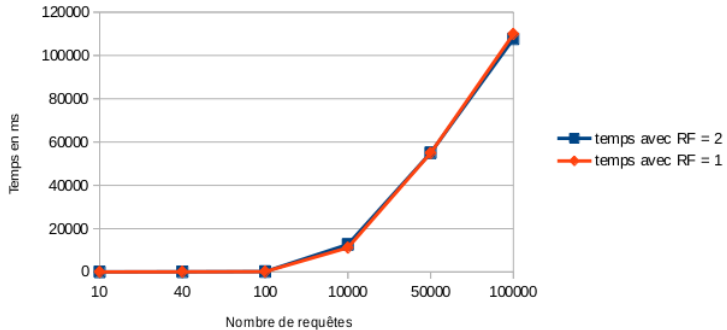
Dénomination

- RF = nombre de copies + donnée originale
- petits objets = un texte généré aléatoirement de 10 Ko
- gros objets = un texte généré aléatoirement de 1 Mo

Tests

Sur Cassandra modifiée

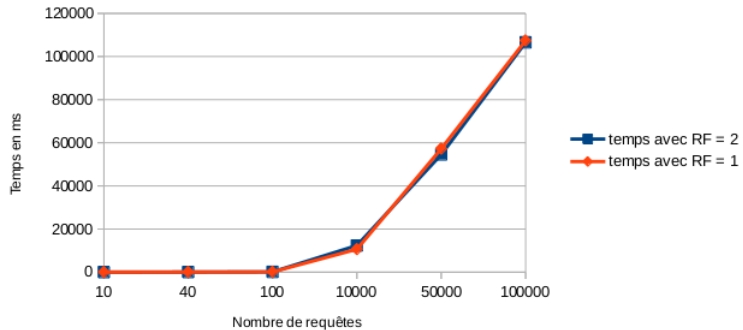
Temps d'exécution sur 10 000 objets de taille 10 Ko en fonction du nombre de requêtes



Tests

Sur Cassandra non modifiée

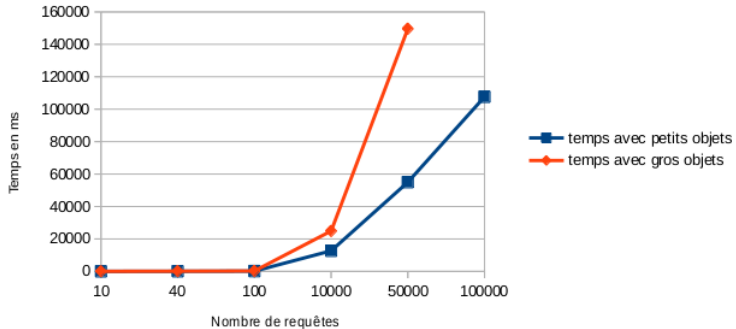
Temps d'exécution sur 10 000 objets de 10 Ko en fonction du nombre de requêtes



Tests

Sur Cassandra modifiée

Temps d'exécution sur 10 000 objets avec RF = 2 en fonction du nombre de requêtes



Perspectives

Cassandra

- Gestion des requêtes
 - Algorithmes de réaffectation SVLO et AverageDegree
- Gestion de la popularité
- Tests unitaires poussés

Application client

-

Visualisation

- V véritable logiciel de vue de performance
- Performance du réseau

Questions ?