Infrastructure de données

M2 - DataScience - Polytechnique

Projet

ESILV
nicolas.travers (at) devinci.fr

Chapitre

Table des matières

Chapitre 1

Modèle de données NoSQL

Le but de ce projet est définir le cadre de conception d'une application Big Data pour une base de données NoSQL orientée document. Pour des raisons d'homogénéisation, nous prendrons comme support MongoDB.

Chaque projet reposera sur un dataset réel trouvé sur le Web qui sera fusionné, partiellement ou intégralement, pour le faire passer à l'échelle sur une infrastructure distribuée.

Un rapport et une présentation finale doit être fournie.

1.1 Choix du jeux de données

Dans le cadre de ce projet, vous devrez choisir votre propre jeu de données. L'idée est de partir d'un jeu de données provenant d'une base de données relationnelle que vous pourrez trouver sur internet, comme celles présentes aux adresses suivantes :

— `https://relational.fit.cvut.cz/search` ;

— `https://toolbox.google.com/datasetsearch`.

Un schéma à plusieurs tables est nécessaire (minimum 4) et si possible un volume de données conséquent.

Chaque jeu de données correspond à un cas d'usage particulier que vous aurez à spécifier pour définir l'infrastructure de données associée.

1.2 Cas d'usage

Pour pouvoir orienter les choix des étapes suivantes, il va falloir étudier les cas d'usage sur votre jeu de données. Nous prendrons deux vues distinctes :

— *End-User view* : Définir, en langage courant, 4 types d'interrogations sur votre jeu de données. On estimera que celles-ci sont effectuées très fréquemment.
Positionnez-vous comme un utilisateur standard de l'application.

— *Data Analyst View* : Définir, en langage courant, 2 types d'interrogations lourdes sur votre jeu de données (agrégation, transformation, calcul complexe).
Positionnez-vous comme un analyste ou un décisionnaire de l'application.

Ces requêtes seront par la suite traduite en MQL (MongoDB Query Language) sur le modèle de données que vous aurez proposé. À cette étape du projet, les requêtes sont simplement un cas d'usage ne dépendant pas de la structure de données que vous produirez.

1.3 Rendu

Pour cette partie, une description du modèle de données d'origine et du cas d'usage pour présenter la problématique doit être présenté pour comprendre ce qui est attendu.

Chapitre 2

Dénormalisation

2.1 Dénormalisation du schéma

Sur ce jeu de données, il va falloir effectuer un choix de dénormalisation pour intégrer les données dans une base de données de type MongoDB. Pour cela, reposez-vous sur les interrogations produites dans la section précédente pour orienter vos choix.

Les points clés de la dénormalisation :

— Reposez vos choix sur les cardinalités entre les tables de votre schéma ;

— Tenez compte les données fréquemment accédées par vos requêtes utilisateurs et le coût élevé de vos requêtes Data Analystes ;

— Ne dénormalisez que ce qui est nécessaire ;

— Tenez compte d'une éventuelle évolution de la volumétrie de votre jeu de données (elle n'est pas statique).

2.2 Types de dénormalisations

Les types de dénormalisation possibles :

— `Fusion` : fusionner deux tables (imbrication, liste)

— `Eclatement` : une table est séparée en deux spécialisations

— `Surcharge` : Une information (attribut) est dupliquée dans une autre table pour éviter un accès inutile

— `Matérialisation` : Le résultat d'un calcul (agrégation) est matérialisé dans un attribut

Vous pouvez user de n'importent quelles étapes de dénormalisation du moment qu'elles soient justifiée.

2.3 Rendu

Les étapes de dénormalisation du schéma et l'argumentaire est nécessaire pour la compréhension des choix effectués.

Une présentation des points clés de la dénormalisation et le schéma obtenu en sortie est attendu. Un exemple de document JSON produit en sortie serait appréciable.

Chapitre 3
Optimisation de l'infrastructure de données

3.1 Requêtes MQL

Reprenez les cas d'usages définies en section 1.2 et traduisez chacun sous forme de requête MQL (MongoDB Query Language). Vous vous aiderez de l'exemple de document JSON produit précédemment pour faciliter l'écriture.

3.2 Choix de clé de sharding

Afin d'optimiser l'infrastructure de données envisagée, il est nécessaire de définir différentes solutions de clé de partitonnement (ou sharding) et d'indexation (locales aux serveurs).

Pour cela, reposez vos choix sur les requêtes produits dans la section 3.1.

3.3 Calcul de coût

Afin de choisir la meilleure combinaison de partitionnement et d'index, produisez un tableau donnant le calcul entre communications effectuées sur le réseau. Nous estimerons que les données sont distribuées sur 100 *shard*.

Exemple :

Requête	Coût Clé 1	Coût Clé 2	...
R_{u1}			
R_{u2}			
R_{u3}			
R_{u4}			
R_{da1}			
R_{da2}			
Total pondéré			

Pour la pondération des requêtes, nous estimerons que :

- R_{u1} : 10000x par jour
- R_{u2} : 1000x par jour
- R_{u3} : 500x par jour
- R_{u4} : 100x par jour
- R_{da1} : 50x par jour
- R_{da2} : 25x par jour

3.4 Rendu

Les requêtes appliquées sur le modèle de données avec rappel du cas d'usage.

Le calcul sous forme de tableau avec le détail des communications réseaux effectuées par chaque requête.