Infrastructure Cloud - Livrables

S9 - Infrastructure de données pour le Cloud

DIA - A5

ESILV
nicolas.travers (at) devinci.fr

Chapitre

Table des matières

Chapitre 1

Modèle de données NoSQL

Le but de ce devoir à la maison 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 final doit être fournie.

1.1 Groupes

Ce travail est à effectuer par groupes de 4 qui pourra choisir son jeu de données qui sera attribué jusqu'à la fin de ce cours.

Les 4 membres du groupe doivent impérativement être dans le même groupe de TD pour pouvoir travailler ensemble lors des séances présentielles.

Pour que le projet soit noté, il est nécessaire d'avoir choisi un groupe. Si vous ne vous êtes pas inscrit à un groupe sur DVO, vous n'aurez donc pas de note.

Quelque soit le nombre de personne du groupe, le projet sera noté de la même manière. La charge de travaille requise est de 10h/personne par rapport. Un rapport devrait prendre normalement 30-40h de travail cumulé par groupe. Si vous êtes seul dans votre groupe, ce sera plus dur pour vous, mais vous n'aurez pas de traitement de faveur - travail d'équipe avant tout.

1.2 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` ;
- `https://www.kaggle.com/datasets` ;
- `https://registry.opendata.aws/` ;
- `https://github.com/awesomedata/awesome-public-datasets#publicdomains` ;
- `http://millionsongdataset.com/` ;
- `https://www.google.com/publicdata/directory` ;
- `https://www.ncdc.noaa.gov/cdo-web/datasets`.

⚠Vous devez respecter les conditions suivantes :

1.2.1 Ne pas prendre le même dataset utilisé dans le cours, celui de l'an dernier, ni celui d'un autre groupe ;

1.2.2 Ne pas copier le travail effectué par un autre groupe/élève ;

1.2.3 Un schéma à plusieurs tables est nécessaire (minimum 4), interconnectées (jointures à faire) et si possible un volume de données conséquent (plus de 400Mo serait bien, sinon minimum 100Mo).

1.2.4 ⚠Certains dataset contiennent plusieurs tables, mais sans jointures. Etudiez bien le contenu du dataset pour faire votre choix.

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

⚠**Pour valider votre choix, vous devrez pour chaque groupe un message dans le fil de discussion en ligne (`https://devinci-online.brightspace.com/d2l/le/65814/discussions/threads/1130/View`), au maximum une semaine après la réception de ce sujet : J+7** (Premier arrivé, premier servi !)

1.3 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. 1 jointure (ou plus) et 1 filtre (ou plus) **minimum** pour chaque requête.

— *Data Analyst View* : Définir, en langage courant, 4 types d'interrogations lourdes sur votre jeu de données (agrégation, jointures, transformations, calcul complexe).
Positionnez-vous comme un analyste ou un décisionnaire de l'application.

La complexité de votre cas d'usage aura un fort impact sur la complexité de votre projet, et de fait sur la note que vous obtiendrez. La variété des jointures est également un critère d'évaluation (éviter de faire toujours la même jointure).

Ces requêtes seront par la suite traduites 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.

1.4 Relier les requêtes au schéma de base de données

Pour chaque requête du cas d'usage :

— Identifier les tables et attributs ciblés ;

— Lister les jointures à effectuer ;

— Lister les filtres sur attributs ;

— Lister les projections ;

— Lister les agrégats, ainsi que les informations nécessaires pour leur application ;

Un schéma récapitutalif appliqué sur le schéma de base de données serait appréciable.

1.5 Statistiques

Sur chaque table, donner l'estimation de :

— Nombre de documents ;

— Cardinalité des attributs associé à un filtre (requête) ;

— Cardinalité des jointures ;

1.6 Rendu

⚠**Envoyer le cas d'usage en même temps que le dataset à J+7**

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.

Noté sur 8 points.

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).

Il est recommandé de produire **2 schémas de base de données différents** (au moins) pour se permettre de faire un choix d'implémentation efficace. Les deux schémas de base de données doivent avoir une structure TRES différentes afin de coller aux différents types de requêtes du cas d'usage. Chaque schéma de bases de données doit lui-même détailler les schémas de chaque collection produite.

Les schémas de collection dénormalisées seront à présenter au format **JSON Schema**. Des jeux de couleurs faciliteront la visualisation des dénormalisations appliquées.

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 Requêtes et statistiques

À partir du rapport précédent, modifier la liste des caractéristiques des requêtes (section 1.4) en fonction de vos dénormalisations.

De même pour les statistiques, mettre à jour le nombre de documents par collection et filtres le cas échéant. De plus, donner le nombre de documents imbriqués dans des listes, le cas échéant.

2.4 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.

⚠**Le rapport doit être rendu à J+21, soit 3 semaines après réception du sujet. Une pénalité de 1 point par jour de retard sera appliqué.**

Noté sur 12 points.

Chapitre 3
Optimisation de l'infrastructure de données

3.1 Requêtes MQL

Reprenez les cas d'usages définies en section 1.3 et traduisez chacun sous forme de requêtes 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 partitionnement (ou sharding) adaptées aux requêtes (clés de filtres ou clés d'agrégations).

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

Vous devrez proposer deux choix de sharding pour pouvoir comparer les coûts.

Les clés filtrées mais non utilisées pour le partitionnement seront associées à un index secondaire.

3.3 Calcul de coût - Modèle de coût réseau

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

Exemple :

Requête	Schéma de base de données 1		Schéma de base de données 2	
	Coût Sharding 1	Coût Sharding 2	Coût Sharding 1	Coût Sharding 2
R_{u1}				
R_{u2}				
R_{u3}				
R_{u4}				
R_{da1}				
R_{da2}				
R_{da3}				
R_{da4}				
Total pondéré				

Pour la pondération des requêtes, nous estimerons que :

- R_{u1} : 10 000x par jour
- R_{u2} : 1 000x par jour
- R_{u3} : 500x par jour
- R_{u4} : 100x par jour
- R_{da1} : 50x par jour
- R_{da2} : 25x par jour
- R_{da3} : 10x par jour
- R_{da4} : 2x 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.

⚠**Le rapport doit être rendu à J+35, soit 5 semaines après réception du sujet. Une pénalité de 1 point par jour de retard sera appliqué.**

Noté sur 10 points.

Chapitre 4

Application Cloud

Vous devez développer une application au dessus de votre BDD dénormalisée. Vous utiliserez le langage de votre choix. Le code source doit être clair et commenté. Différentes parties doivent être clairement identifiées :

- La connexion à MongoDB ;
- Importation de votre dataset respectant le schéma de dénormalisation ;
- Mesures de performances ;
- Les vues.

4.1 Importation des données

Afin de dénormaliser votre dataset en respectant le schéma produit dans le 2ème livrable. Pour ce faire, plusieurs stratégies sont possibles :

- Développer un script de lecture des CSV pour produire les documents JSON adaptés ;
- Intégrer chaque CSV dans MongoDB puis de faire la conversion avec des requêtes MongoDB (opérateur $lookup, utilisable en local, pas en sharding) ;
- Utiliser l'outil Studio3t en version professionnelle (demander une licence en amont) ;
- Trouver un logiciel de transformation...

4.2 Mesure de performances

Afin d'évaluation l'efficacité de votre solution, vous devrez développement un programme de test pour chaque requête de votre cas d'usage. Celui-ci devra pouvoir fournir :

- Execution de chaque requête ;
- Mesurer le temps d'exécution de chaque. Pour cela, la requête doit être exécutée sur l'ensemble de la BDD, au moins 10 fois (retirer le max et le min), et faire la moyenne. Le temps de récupération des données n'est pas comptabilisé.
- Mesurer le temps sur différentes configuration de sharding : de 1 à 6 shards(minimum) afin de mesurer l'impact de la distribution
- Le rapport doit pouvoir mettre en valeur les performances liées à chaque requête, et expliquer les variations obtenues (explication $\neq$ observation).

Noté sur 10 points.

4.3 Vues

Vous devrez définir des vues en fonction des requêtes définies précédemment.

4.3.1 Utilisateur standard

Fournir à un utilisateur standard un accès aux résultats des 4 requêtes simples. La vue n'a pas forcément besoin de fournir la requête mais seulement l'interaction avec elles et une vision sur les données interrogées (ou visulation si possible).

4.3.2 Analyste/Décisionnaire

Fournir à votre Data Analyst ou votre Business User une vue permettant de fournir les requêtes complexes proposées. L'interface devra permettre de paramétrer les requêtes avec des valeurs (soit à la main, soit par menu déroulant).

Une DataViz est fortement recommandée pour les résultats obtenus.

4.3.3 Administrateur

L'administrateur de la base MongoDB doit pouvoir récupérer différentes statistiques pour pouvoir faire évoluer le cluster en fonction de la charge. La vue doit pouvoir fournir les informations suivantes :

— Les statistiques de distribution des données ;

— Consulter l'état du cluster : nombre de shards, nombre de réplicats par shard ;

— Répartition des données sur les shards (nombre de documents) ;

— Indexes existants sur les données.

4.4 Rendu

Un rapport donnant les mesures de performances de vos requêtes, une description de l'architecture de votre application et serveurs (MongoDB), une description globale de l'organisation de votre code. Egalement un zip ou Git pour consulter votre code.

⚠**Le rapport doit être rendu à J+56, soit 8 semaines après réception du sujet. Une pénalité de 1 point par jour de retard sera appliqué.**

Noté sur 10 points.