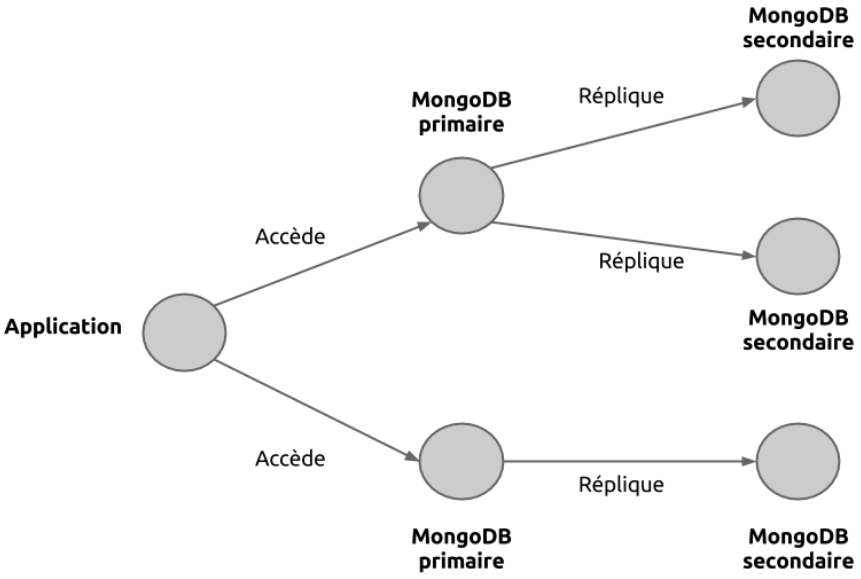
Module Nouvelles Technologies du Web

TME - Développement et réplication avec MongoDB

I°) Problématique de la séance

Le BigData est une des nouvelles problématiques de tous les jours, nous stockons toujours plus de données. Des technologies émergentes, comme Hadoop, ont apportées des solutions à la gestion de gros volumes de données. Un fichier est découpé en chunks de taille fixe, répartis sur différentes machines d'un cluster. L'objectif de la séance sera la gestion de chunks avec MongoDB.



L'API réalisée permettra de propager des commandes à des serveurs MongoDB de manière transparente pour le développeur. Vous devrez implémentée deux types de stockage pour les chunks,

dans des collections standards, avec l'API MongoDB classique, et avec GridFS. De plus, vous aurez un aperçu du design pattern composite et des stratégies de placement de chunk sur les serveurs primaires.

Documentation MongoDB : http://www.mongodb.org/display/DOCS/Home
JavaDoc du driver MongoDB : http://api.mongodb.org/java/current/
Tutoriel Java officiel pour MongoDB : http://www.mongodb.org/display/DOCS/Java+Tutorial

Prérequis :

1. Eclipse 4.2 Juno
2. Java 7
3. JUnit 4
4. MongoDB

II°) Environnement de travail

Télécharger le projet Java à l'adresse suivante :
https://github.com/olivier-pitton/master/archive/dant.zip

Dézippez le fichier et récupérer l'archive "dant-mongo.tar.gz", dans le dossier mongodb.

Dézippez le projet, dans le tar.gz, dans votre workspace.

Dans Eclipse, faites Clic droit (dans *Package Explorer*) -> *Import project* et sélectionnez le projet.

Une fois le projet importé, vous pourrez démarrer le TP.
Le projet est structuré de la manière suivante :

1. *src* : Contient les sources du projet. Chaque classe développée devra se trouver dans le package adéquat.
2. *test* : Contient une batterie de tests pour valider ce que vous avez réalisé. Ces tests s'appuient sur la bibliothèque JUnit 4. Si vous désirez lancer la totalité des tests, vous pouvez utiliser la classe *TestSuite*. Les tests se lancent par défaut sur localhost sur les ports 27017 et 27018. Donc pensez à lancer deux serveurs.

Lors du rendu, vous devrez impérativement laisser la configuration telle qu'elle était.

III°) Développement de l'API

Avant de poser des questions, pensez à lire la documentation présente dans les interfaces dont vous réaliserez les implémentations.

3.1°) Création de l'objet métier

Créer une classe ChunkImpl implémentant l'interface Chunk. L'unicité d'un chunk se fait par son nom et son id. Par exemple si le fichier toto.txt est envoyée à l'API, il sera splitté en N chunks ayant comme max N, comme nom "toto.txt" et comme id la position dans le fichier (donc le début a l'id 1 et le dernier N).

Pensez à redéfinir les méthodes equals / hashcode (vous pouvez les générer avec Eclipse). La

méthode toString() de votre implémentation doit renvoyer une chaîne formatée comme suit : <nom>@<id>. Exemple “toto.txt@2” pour le fragment 2 du fichier toto.txt.

Mettez à jour la classe ChunkFactory afin qu’elle puisse créer vos chunks.

Lancez le test ChunkTest pour valider votre travail.

3.2°) Implémentation standard avec MongoDB

Créez une classe ChunkStorageImpl héritant de la classe AbstractChunkStorage. Cette classe a pour but d’offrir des méthodes standards pour communiquer avec MongoDB. Pas de GridFS, simplement des insertions dans les collections. Tout ChunkStorage a état, démarré ou arrêté. Cette implémentation ne possède qu’une unique connexion à MongoDB (donc le premier élément du paramètre dans la méthode start).
Lisez attentivement les commentaires de l’interface ChunkStorage pour réaliser cette classe. Des méthodes de conversion entre un Chunk et un DBObject sont présentes dans AbstractChunkStorage.

Mettez à jour la factory ChunkStorageFactory.

Lancez le test unitaire ChunkStorageImplTest pour valider votre travail.

3.3°) Implémentation de GridFS

Créez une classe ChunkStorageFS héritant de la classe AbstractChunkStorage. Cette classe effectue les mêmes opérations que l’implémentation standard, mais en utilisant GridFS. De la même manière, on ne se connecte qu’à un unique serveur.

Mettez à jour la factory ChunkStorageFactory.

Lancez le test unitaire ChunkFSStorageTest pour valider votre travail.

3.4.1°) Stratégies de placement

Créez deux classes HashStrategy et RoundRobinStrategy implémentant l’interface ChunkStrategy. Ces classes permettront de définir des stratégies de placement des chunks. Le premier hache le nom du chunk modulo le nombre de serveurs pour obtenir le serveur où mettre / rechercher le fragment. La seconde est simplement une implémentation de l’algorithme du tourniquet : On place le premier fragment sur le serveur 0, le second sur le 1, … modulo le nombre de serveurs.

Mettez à jour la factory ChunkStrategyFactory.

Lancez les tests unitaires HashStrategyTest RoundRobinStrategyTest pour valider votre travail.

3.4.2°) Introduction au pattern Composite

Créez une classe ChunkStorageComposite héritant de la classe AbstractChunkStorage. Cette classe possédera une liste de ChunkStorage et une stratégie. Tous les serveurs donnés lors de l’appel à start devront être enregistrés dans la liste.

Lors d’une opération d’écriture, la stratégie sélectionne le serveur sur lequel écrire.

Lors d'une opération de lecture, si la stratégie est aléatoire, on recherche sur tous les serveurs, sinon on utilise la stratégie pour retrouver le bon serveur.

Mettez à jour la factory ChunkStorageFactory.

Lancez le test unitaire ChunkStorageCompositeTest pour valider votre travail.

3.5°) Mise en place d'un mécanisme de réplication (optionnel)

Lancez quatre serveurs sur votre machine de telle manière que deux serveurs soient maître de deux serveurs esclaves. Lors d'une écriture sur un serveur maître celui-ci réplique sur son esclave.

Les deux serveurs maîtres doivent être lancés sur les ports 27017 et 27018.

Faites valider votre travail par l'encadrant.

3.6°) Optimisation de l'architecture (optionnel)

Notre système comprend 10 fois plus de lectures que d'écritures et ne doit pas perdre de données, en aucun cas.

Comment optimiser notre modèle en prenant cela en considération ?

Faites valider votre travail par l'encadrant.