TD Base de données distribuée avec MongoDB

Objectif
Découvrir la base de données NoSQL MongoDB et son modèle de réplication. Elle appartient à la catégorie des bases orientées documents, est développé par la société américaine 10Gen en C++.

Installation
Sur le site mongodb.com, téléchargez la version community server (version supérieur à 4.4). Attention à prendre une version qui correspond à votre distribution (on peut avoir de mauvaise surprise avec le premier choix de la liste : Amazon Linux). Poursuivre l'installation en dézippant le fichier tgz. Vous devez avoir un répertoire du type (pour la version 4.4.1) :

```
mongodb-linux-x86_64-ubuntu2004-4.4.1
```

Terminologie
Un document est l'unité de base de données. Similaire à un tuple.
Une collection peut être considérée comme une table dans le contexte SGBDR mais sans schéma et représentée en JSON.
Une base de données est un ensemble de collections.

Lancement
Dans le répertoire **bin** de votre répertoire mongodb, on trouve plusieurs fichiers qui exécutables que nous utiliserons pendant cette session sur MongoDB. Le mode d'exécution de MongoDB est du type client/serveur. Par défaut, le serveur écoute sur le port 27017 (et sur 28017 pour des informations adminstratives).
Pour que le lancement fonctionne, il est nécessaire de créer un répertoire (par ex. data dans votre répertoire courant - celui de votre installation par exemple ou dans le répertoire /tmp sur les machines de l'université). On lance le serveur avec l'exécutable (mongo daemon) en indiquant le répertoire où se trouveront les données

```
mongod --dbpath repCourant/data
```

Depuis une autre console, le shell de MongoDB se lance avec l'exécutable `mongo`. C'est un shell javascript (donc interprétation du code possible dans ce langage de programmation) qui permet d'administrer la base et de lancer des requêtes.
Pour faire quelques rapides exemples dans la console:

```
>x=20
>x/4
5
> function facto(n) { if(n<=1) return 1;
... return n * facto(n-1);
... }
> facto(5)
120
```

Shell

La commande `help` permet d'obtenir une liste des commandes disponibles : `show dbs`, `show collections`, `use <dbName>` (pour indiquer la base avec laquelle on souhaite travailler).

JSON

Le système de gestion de bases de données NoSQL MongoDB gère des documents JSON (JavaScript Object Notation). C'est un format de données textuelles provenant de la notation objet du langage Javascript consistant en des paires de clé/valeur et des tableaux.

Soit l'exemple de document JSON suivant :

```
{
     "nom" : "Davis",
     "prenom" : "Miles",
     "age" : 65,
     "adresse" : {
          "city" : "NY",
          "zip" : "10021"
     },
     "phones" : [
          {
               "type" : "home",
               "phone" : "111-555-1234"
          },
          {
               "type" : "mobile",
               "phone" : "116-555-1234"
          }
     ]
}
```

Requêtage

1. Vous allez travailler avec la base de données testm. Depuis cette base, vous allez enregistrer le document précédent dans la collection pers. Cette insertion se fait à partir de l'instruction suivante :

`db.nomCollection.save(documentJson)` db et save sont des mots clés de mongdb.

Vous devriez avoir un retour sur la console du type :

```
WriteResult({ "nInserted" : 1 })
```

2. On veut maintenant lire le contenu de la collection `pers`. Pour cela, on faire saisir `db.nomCollection.` (donc dans notre cas `db.pers.`) puis on va appuyer sur la touche d'indentation pour voir par autocomplétion la liste des opérations possibles sur une collection.
Dans un premier temps, saisisser findOne(). On observe qu'un couple clé/valeur a été ajouté :

"_id" : ObjectId("5ba9def22b9ab6f623ee59d8")

Dans votre cas, la valeur dans ObjectId sera vraisemblablement différente de la valeur ci-dessous.

La clé '_id' correspond à une clé primaire pour chaque document. Elle est générée automatiquement par mongodb mais peut aussi être renseignée par l'utilisateur (qui dans ce cas doit garantir de son unicité).
Pour pouvez tester les opérations findOne(), count()

3. On va expérimenter avec un jeu de données plus important. Vous allez charger un jeu de données sur des films et séries (films.json). A partir de la version 4.4, l'exécutable mongoimport ainsi que d'autres exécutables ne sont plus intégrés dans la version Server. Il faut télécharger le MongoDB Database Tools depuis la page https://www.mongodb.com/try/download/database-tools. Vous pouvez maintenant charger le jeu de données à l'aide de l'éxécutable mongoimport. Regardez la page https://docs.mongodb.com/manual/reference/program/mongoimport/ pour sélectionner les bons arguments à utiliser avec ce fichier json.
Vous devez avoir 16 documents dans la collection de votre base de données.

Rédiger les requêtes suivantes en vous aidant de la page suivante
https://docs.mongodb.com/manual/crud/

4. Afficher les titres (Title) des documents de la collection films

5. Afficher toutes les données des documents qui sont des films (type est movie)

6. Combien y-a-t-il d'entrées dans la requête précédente. Est-ce que toutes les entrées de la base sont des films.

7. Afficher les types (« Type ») de documents de la collection films.

8. Afficher les documents qui sont des films (movies) de 2016.

9. Modifier le requête 8 pour n'avoir que le nom du film et son année dans le résultat (on ne veut pas avoir _id)

Réplication
Le mode de réplication dans mongoDB est basé sur la notion de **replica set** qui correspond à un ensemble de processus mongod. C'est donc un cluster Master-Slave avec une gestion de dépannage automatique (automatic failover). C'est-à-dire que le replica set va élire un master sur le cluster mais si celui-ci tombe en panne, une autre machine du cluster prendra sa place (ce mécanisme n'est pas disponible dans l'approche Master-Slave classique).
Terminologie : le master est le primary node et les slaves sont des secondary nodes. La configuration d'un replica set utilise l'exécutable mongod mais nécessite des arguments supplémentaires :

- `--replSet nomDuReplicaSet`. On donne un nom au replica set (par exemple rep)
- `-- port numeroDuPort`. On prendre par exemple le port 27017

10. Comme dans notre exemple avec un unique serveur, il faut créer les répertoires contenant les données (mkdir -p dbs/rs1). On considère que vous avez 3 répertoires (tous dans dbs) : rs1, rs2 et rs3
Exemple de lancement d'un daemon avec le port 27017

```
bin/mongod --replSet rep --dbpath ~/dbs/rs1 --port 27017
```
Vous devez créer les 2 autres répliques avec respectivement rs2/27018 et rs3/27019 (sur le même replSet rep)

11. Dans une nouvelle console, vous allez lancer le shell mongo sur le port 27017 :
```
mongo --port 27017
```
Pour initialisation le mode replica set de mongodb. Pour cela, on va saisir la variable JSON suivante :
```
config = {
 _id: "rep",
 members: [
 {"_id": 0, "host": "localhost:27017"},
 {"_id": 1, "host": "localhost:27018"},
 {"_id": 2, "host": "localhost:27019"}
 ]
}
```
On voit que le replica set à un _id et que chaque membre à également un _id et un host (correspond aux numéros de port utilisés lors du lancement des daemons).

Maintenant, on initialise le replica set avec l'instruction :
```
rs.initiate(config)
```

12. A l'aide du préfixe « rs », on peut accéder à des informations sur le replica set. Regarder les fonctions possibles à partir de l'autocomplétion sur rs.

On va lancer un rs.status() dans notre console shell. Il y a beaucoup d'informations. On peut y voir en particulier des informations sur les membres du replica set, dont l'état.
Identifier le primaire (master) et les secondaires (esclaves) du replica set.

Pour vérifier, faites un rs.isMaster() sur a console shell actuelle.

13. On va lancer des écritures sur le master (notre unique console avec un shell pour le moment). Sélectionner une base de données (par exemple db1).
```
for(i=0;i<1000;i++){db.col2.insertOne({"count":i})}
```

Vérifier le nombre de documents dans la collection col2.
Dans une nouvelle console, lancer un shell mongo sur le port 27018. On ignore les warning. A l'invitation du shell, vous devez voir si 27018 est un primary ou secondary sans faire un rs.isMaster() ou rs.status() (cf. l'invitation de la console).
On veut voir les bd disponibles sur 27018 : show dbs
On se retrouve avec un message d'erreur. En fait, un secondaire peut être en retard par rapport à son primaire (il n'a pas reçu toutes les données du primaire). Par défaut, les lectures sont refusées sur les secondaires. Pour permettre l'exploitation d'un secondaire, il faut saisir l'instruction :
```
rs.secondaryOk()
```

On relance `show dbs`. On voit la base db1.
Ensuite, on sélectionne db1 et on demande le nombre de documents dans sa collection col2.

Pour observer la synchronisation entre le master et un secondaire, on va ajouter un nouveau document depuis la console du master. Vous devez avoir 1001 documents dans col2 maintenant. Faire rapidement un count sur cette collection dans la console de 27018.

15. Et si on veut faire un insertOne depuis 27018, à votre avis que se passera-t-il ?

16. On va tester la réélection automatique d'un master sur notre cluster de 3 répliques.
Depuis le master :

```
db.adminCommand({"shutdown":1})
```

Essayez un `rs.status()` sur le cette console. Surpris ?

Un `rs.status()` depuis les consoles de 27018 et 27019. Observer les invitations des consoles. Vérifier que vous pouvez faire des écritures sur le master, pas sur le secondaire.

17. Et si démarrait 27017 ? Deviendrait-il à nouveau le master du replica set rep ?