MIF04 GDW – TP
Map/Reduce sur MongoDB – partie 1

Résumé

Cette première partie du TP a le double objectif de découvrir MongoDB et de mettre en œuvre des traitements Map/Reduce en `javascript`. La seconde partie du TP, dans un autre sujet, sera à rendre.

1 Environnement de travail

1.1 Documentation de référence

— http://docs.mongodb.org/manual/core/map-reduce/
— http://docs.mongodb.org/manual/reference/mongo-shell/
— https://docs.mongodb.com/manual/reference/method/db.collection.mapReduce/#db.collection.mapReduce

1.2 Serveur

Le serveur MongoDB déployé sur la plate-forme de cloud [1] est accessibles à l'adresse IP `192.168.74.37:27017`. Vous utiliserez le client en ligne de commande `mongo` depuis Linux [2]. Vous vous authentifierez avec l'utilisateur `mif04` et le mot de passe `m2i0f1084`. Vous travaillerez sur les collections de la base de données `mif04`. Pour travailler sur votre propre serveur, vous pouvez télécharger les jeux de données et les importer avec les commandes suivantes :

```
wget http://media.mongodb.org/zips.json
wget https://raw.githubusercontent.com/ozlerhakan/mongodb-json-files/master/datasets/grades
wget https://raw.githubusercontent.com/mongodb/docs-assets/geospatial/restaurants.json
wget https://raw.githubusercontent.com/mongodb/docs-assets/geospatial/neighborhoods.json

mongoimport --host localhost  --db mif04 --collection zips --drop --file zips.json
mongoimport --host localhost  --db mif04 --collection grades --drop --file grades.json
mongoimport --host localhost  --db mif04 --collection restaurants --drop --file restaurants
mongoimport --host localhost  --db mif04 --collection neighborhoods --drop --file neighborh
```

2 Prise en main

Identifiez vous avec la commande suivante :

```
> mongo -u "mif04" -p "m2i0f1084"  --authenticationDatabase "mif04" 192.168.74.37/mif04
```

1. accès depuis le campus uniquement, utiliser impérativement `eduroam` en wifi ou un tunnel `ssh` pour un accès depuis l'extérieur.
2. Pour l'installation https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/

Exercice 1 : premiers pas en MongoDB

1. Exécutez les commande `show collections` et `show dbs`. Que font elles ?
2. Trouvez la commande MongoDB pour afficher *un document* de la collection `zips`. Expliquer la structure du document trouvé.
3. Trouvez la commande MongoDB pour calculer *le nombre de documents* de la collection `zips`.
4. Reprendre les questions précédentes avec les collections `grades` et `restaurants`.
5. Trouvez la commande MongoDB qui permet de donner la liste des *zips* de l'état de Californie.
6. Même question que précédemment, mais cette fois vous ne voulez garder dans le résultat que le nom de la ville et *aucun autre champ*.
7. Trouvez la commande MongoDB qui permet de lister les *zips* dont la population est supérieure à 100.000 habitants. Même question avec *le nombre* de *zips* dont la population est supérieure à 100.000 habitants.

3 Découverte de Map/Reduce

Dans toute la suite de ce TP, vous devrez répondre aux exercices avec des requêtes Map/Reduce. Vous n'avez pas les droits d'écriture sur le serveur, il faut donc afficher le résultat de vos jobs dans la console et vous devrez résoudre chaque exercice *avec un seul job*.

Télécharger le fichier `MIF04-GDW-TPMongoDB-Init.js`[3] et consultez le. Ensuite, chargez le fichier avec la commande suivante :

```
> load("MIF04-GDW-TPMongoDB-1-Init.js");
```

Exercice 2 : premier comptage en Map/Reduce

Exécutez la commande `db.zips.mapReduce(init_map, init_red, {out : {inline:1}});`.

1. Expliquez ce que fait la ligne `emit(0, 1);`. Le résultat est-il le même si on la remplace par `emit("obiwan kenobi", 1);` ? Justifier. Que conseilleriez vous au final comme constante à choisir ?
2. Qu'est-ce que ce job Map/Reduce calcule ? Exécutez ce job sur d'autres collections et vérifiez le résultat.
3. Que contient le champs `counts` de l'objet retourné ? Expliquez.
4. Exécutez maintenant `db.zips.mapReduce(init_map, bogus_red, {out : {inline:1}});`. Quel est le problème ? Justifiez[4].
5. Écrire un job Map/Reduce qui, pour chaque état des USA, calcule sa population. Vérifiez la cohérence du résultat, sur par exemple la population de la Californie.
6. Écrire un job Map/Reduce comme précédemment qui, pour chaque état des USA, donne le nombre de *zips* dont la population est supérieur à 100.000.
7. Reprendre la question précédente en utilisant le champ `query` de `mapReduce`. Que constatez vous ? Expliquez.

3. https://perso.liris.cnrs.fr/romuald.thion/files/Enseignement/MIF04/MIF04-GDW-TPMongoDB-Init.js
4. https://docs.mongodb.com/manual/reference/method/db.collection.mapReduce/#requirements-for-the-reduce-function

Corrections

Solution de l'exercice 1

1. Affiche la liste des collections disponibles pour cet utilisateur.

```
> show dbs
mif04  0.005GB
test   0.000GB

> show collections
grades
neighborhoods
restaurants
zips
```

2. `db.zips.findOne();`

```
> db.zips.findOne();
{
"_id" : "01001",
"city" : "AGAWAM",
"loc" : [
-72.622739,
42.070206
],
"pop" : 15338,
"state" : "MA"
}
```

3. `db.zips.count();`

```
> db.zips.count()
29353
```

4. Trivial

5. `db.zips.find({state:"CA"});`

```
> db.zips.find({"state" : "CA"}).pretty()
{
"_id" : "90006",
"city" : "LOS ANGELES",
"loc" : [
-118.291687,
34.049323
],
"pop" : 63389,
"state" : "CA"
}
...
```

6. `db.zips.find({state:"CA"}, {_id: 0, city:1});`

```
> db.zips.find({state:"CA"}, {_id: 0, city:1});
{ "city" : "LOS ANGELES" }
{ "city" : "LOS ANGELES" }
...
```

Ou encore mieux `db.zips.distinct("city",{state:"CA"});`

```
> db.zips.distinct("city",{state:"CA"});
[
"LOS ANGELES",
"EAST LOS ANGELES",
"CITY OF COMMERCE",
"COLE",
"VERNON",
...
]
```

7. `db.zips.find({pop : {$gte : 100000}});` et `db.zips.count({pop : {$gte : 100000}});`

```
> db.zips.find({pop : {$gte : 100000}}).pretty();
{
"_id" : "10025",
"city" : "NEW YORK",
"loc" : [
-73.968312,
40.797466
],
"pop" : 100027,
"state" : "NY"
}
{
"_id" : "10021",
"city" : "NEW YORK",
"loc" : [
-73.958805,
40.768476
],
"pop" : 106564,
"state" : "NY"
}
{
"_id" : "11226",
"city" : "BROOKLYN",
"loc" : [
-73.956985,
40.646694
],
"pop" : 111396,
"state" : "NY"
}
{
"_id" : "60623",
"city" : "CHICAGO",
"loc" : [
-87.7157,
41.849015
],
"pop" : 112047,
```

```
"state" : "IL"
}

> db.zips.count({pop : {$gte : 100000}})
4

> db.zips.find({pop : {$gte : 100000}}).count();
4
```

Solution de l'exercice 2

1. Elle émet un résultat pour le `map`. Le premier champs de `emit` c'est la clef et le second la valeur associée, on met donc ce qu'on veut ici pour l'unique clef. Une constante explicite comme `emit('nb_documents', 1);` serait conseillée.
2. La même chose que `db.zips.count();`.
3. Des statistiques sur l'exécution, respectivement :
   — `"input" : 29353` le nombre d'objets lus
   — `"emit" : 29353` le nombre de paires émises par `map`
   — `"reduce" : 294` le nombre de fois où `reduce` a été appelée. *Attention* : on voit ici 294 appel et pas un seul appel, c'est les *re-reduces* de MongoDB qui va exécuter `reduce` dès que 100 valeurs ont été produites ! Voir la question suivante.
   — `"output" : 1` est-ce que la sortie est dans une collection ou dans le terminal
4. Le problème c'est celui du *re-reduce* : le `reduce` n'attend pas que `map` ait fini est et appelé sur des sous ensembles du résultats du `map`, on en en trouve ici 294 alors qu'en Hadoop on en aurait un seul. Avec `bogus_red`, le problème est qu'on a eu des agrégations intermédiaires et calculer seulement la longueur n'est plus équivalent à faire la somme. On voit qu'au dernier appel `reduce` a agrégé 5 valeurs.
5. et suivantes : voir le fichier de réponse

```
/*************************************************************/
var ex2_map = function () {
  emit(this.state, this.pop);
}
var ex2_red = function (key, values) {
  return Array.sum(values);
}
//load("../reponses/MIF04-GDW-TPMongoDB-1-Ex1.js");
//db.zips.mapReduce(ex2_map, ex2_red, {out : {inline:1}});

/*************************************************************/
var ex3_map = function () {
  if (this.pop >= 100000)
    emit(this.state, 1);
}
var ex3_red = ex2_red;
// db.zips.mapReduce(ex3_map, ex3_red, {out : {inline:1}});

var ex3_map2 = function () {
  emit(this.state, 1);
}
// db.zips.mapReduce(ex3_map2, ex3_red, {out : {inline:1}, query : {pop :
    {$gte : 100000}}});
// la différence de perf est assez claire : ~300ms pour la première et ~80
    ms pour la suivante
```