MIF18 BDAV – TP
Map/Reduce sur MongoDB

Résumé

L'objectif de ce TP est de mettre en œuvre des traitements Map/Reduce sur MongoDB. Le TP est à réaliser seul ou en binôme, de préférence sous Linux, et sera rendu au plus tard le *17 novembre 2015 à 18h* sur le dépôt Spiral *TP-MongoDB-2015* http://spiralconnect.univ-lyon1.fr/webapp/activities/activities.jsp?containerId=5059007.

Vous devrez rendre une archive `zip` nommée `p0123456_p6543210.zip` où `p0123456` et `p6543210` sont les identifiants Lyon 1 des membres du binôme.

La correction étant automatisée, vous devrez *impérativement* respecter le nom de l'archive, les prototypes et les identifiants des fonctions fournies ainsi que la structure des résultats attendus tels que précisés dans l'archive de départ http://liris.cnrs.fr/romuald.thion/files/Enseignement/MIF18/MIF18-2015-Mongo-reponses.zip.

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/
— http://docs.mongodb.org/manual/reference/command/mapReduce/
— http://stackoverflow.com/questions/6333164/how-do-i-use-map-reduce-in-mongodb
— http://www.w3schools.com/js/default.asp

1.2 Serveurs

Il y a au total 4 serveurs MongoDB déployés sur la plate-forme de cloud [1], configurés et accessibles aux adresses IPs suivantes sur le port 8080. Pour équilibrer les charges, vous travaillerez avec le serveur correspondant *au dernier chiffre du numéro d'étudiant du plus jeune étudiant du binôme modulo 4*, e.g. *MARTIN Jacques 11312386* serveur numéro 2.

serveur 0 `192.168.74.170:8080`

serveur 1 `192.168.74.171:8080`

serveur 2 `192.168.74.172:8080`

serveur 3 `192.168.74.173:8080`

1.3 Collections

Vous utiliserez le client en ligne de commande `mongo` depuis Linux [2]. Vous vous authentifierez avec l'utilisateur `mif18` et le mot de passe `MaIbFc1d8`. Les collections suivantes sont configurées dans la base de données `mif18` :

— `restaurants` : une collection d'évaluations de restaurants aux États-Unis ;

1. accès depuis le campus uniquement, utiliser un tunnel `ssh` pour un accès depuis l'extérieur.
2. package Ubuntu `mongodb-clients`

— `zips` : une collection de données sur les code postaux aux États-Unis ;
— `states` : les états des États-Unis et leur population d'après la collection `zips` ;

Pour travailler sur votre propre serveur, vous pouvez récupérer et importer ces jeux de données avec les commandes suivantes :

```
wget http://media.mongodb.org/zips.json
mongoimport --db mif18 --collection zips --drop --file zips.json

wget https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/dataset.json
mv dataset.json restaurants.json
mongoimport --db mif18 --collection restaurants --drop --file restaurants.json
```

2 Prise en main

```
> mongo 192.168.74.XXX:8080/mif18 --username "mif18" --password "MaIbFc1d8"
...
> db.auth("MIF18-MongoDB","MaIbFc1d8");
1
> show collections;
restaurants
system.indexes
zips
zola
> db.zips.count();
29353
> db.restaurants.count();
25359
```

FIGURE 1 – script d'initialisation

Exercice 1 : chargement de fichier

1. Exécutez les commandes du script d'initialisation (FIGURE 1), vérifiez que vous avez les mêmes valeurs puis exécutez la commande `db.zips.findOne();`. Expliquez ce qui est affiché.
2. Créez un fichier `.js` contenant la définition `var exo1 = 'Hello World!';` puis chargez le depuis le shell de MONGODB. Affichez le contenu de la variable `exo1` dans le shell.

Exercice 2 : commandes de base

Le fichier de réponses pour cet exercice est nommé `MIF18-TP-MapReduce-Ex2.js`.

Avant toute chose, remplissez les variables `etu1Nom`, `etu1Prenom`, `etu1Num`, `etu2Nom`, `etu2Prenom`, `etu2Prenom` avec les noms, prénoms et numéro des étudiants du binôme (laissez le valeurs par défaut pour le deuxième si monôme). Ensuite, en utilisant les commandes de base du *shell* MONGODB, affectez les variables demandées :

1. Affectez le nombre total d'enregistrements dans la collection `zips` à la variable `exo2q1`.
2. Affectez le nombre code postaux correspondant à une ville nommée `SPRINGFIELD` à la variable `exo2q2`.

3. Affectez le nombre d'états aux US à la variable `exo2q3`. Vérifiez que votre résultat est cohérent avec la réalité.
4. Affectez la liste codes postaux dont la population est supérieure à 100.000 personnes à la variable `exo2q4`. On fera une projection pour ne garder que le code postal et l'état correspondant.

Exercice 3 : premiers jobs Map/reduce

Le fichier de réponses pour cet exercice est nommé `MIF18-TP-MapReduce-Ex3.js`.

1. Calculer le nombre total d'enregistrements de la collection *zips* en remplissant les fonctions `exo3q1map` et `exo3q1red`. Le résultat du job Map/Reduce sera stocké dans la variable `exo3q1`.
2. Modifier la fonction `exo3q1red` pour utiliser la fonction `values.reduce()` à la place de la boucle et l'enregistrer dans la variable `exo3q2red`.
3. Modifier la fonction `exo3q1red` pour utiliser la fonction `values.forEach` à la place de la boucle et l'enregistrer dans la variable `exo3q3red`.

3 Pipeline Map/Reduce sur MongoDB

Exercice 4 : requêtes en Map/Reduce

Le fichier de réponses pour cet exercice est nommé `MIF18-TP-MapReduce-Ex4.js`. Vous donnerez les définitions des variables demandées correspondant aux jobs Map/Reduce des questions suivantes.

1. Donner les codes postaux et leur population quand elle est supérieure à 100.000 en filtrant dans le `map`.
2. Même question mais en filtrant avec le paramètre `query` de la fonction `mapReduce()`[3]. Laquelle des deux solutions faut-il privilégier ?
3. Donner la population totale de chaque état, puis en `javascript` avec `sort` et `slice`, filtrer ce résultat en `javascript` pour ne garder que les 3 états les plus peuplés. Vérifiez que votre résultat est cohérent avec la réalité.
4. Donner la liste des villes des US, attention à ne pas mélanger *des villes homonymes mais dans des états différents*. Pour tester votre requête, utiliser le paramètre `limit` de `mapReduce()`.

Exercice 5 : requêtes avancées en Map/reduce

Le fichier de réponses pour cet exercice est nommé `MIF18-TP-MapReduce-Ex5.js`. Vous donnerez les définitions des variables demandées correspondant aux jobs Map/Reduce des questions suivantes.

1. Pour chaque état, donne la liste de ses villes sans supprimer les doublons. *Aide : il faut impérativement que le type des valeurs émises par le map (retourné par `emit`) soit le même que le type d'entrée des valeurs du reduce (paramètre `values` de la fonction) et le type de sortie du reduce (retourné par `return`) soient les mêmes*[4][5].

3. http://docs.mongodb.org/manual/reference/operator/query/
4. http://isurues.wordpress.com/2013/05/28/what-is-re-reduce-in-mongodb-map-reduce/
5. http://stackoverflow.com/questions/3837394/mongodb-map-reduce-over-multiple-collections

2. Même question, mais en supprimant les doublons. *Aide : utiliser un objet `json` comme un tableau associatif pour garantir l'unicité.*
3. Pour chaque état, donner à la fois le code postal le plus peuplé et le moins peuplé. *Aide : utiliser un objet `json` comme valeur du `emit`.*
4. Donner l'état le plus peuplé des US un job Map/Reduce sur la collection `states`.
5. Même question avec un seul job directement depuis la collection `zips`. *Aide : tout le calcul se passe désormais dans le `reduce` et dans `finalize`.*
6. Donner la population moyenne des codes postaux de chaque état. *Aide utilisez la fonction `finalize(key, val)`*[6]

4 Aggregation Pipeline et performance sur MongoDB

Exercice 6 : comparaison de performance

Le fichier de réponses pour cet exercice est nommé `MIF18-TP-MapReduce-Ex6.js`.

MongoDB dispose d'une alternative à Map/Reduce appelée *aggregation pipeline* pour l'évaluation de requêtes[7]. On considère la requête *donner la population totale de chaque ville* dont on va comparer la vitesse d'exécution selon les deux approches.

1. Écrire cette requête en utilisant l'*aggregation pipeline*[8].
2. Comparer la vitesse d'exécution entre les deux pipelines sur la collection `zips`. On ne demander pas d'afficher les résultats mais de les stocker dans des variables. Pour une meilleure précisions, lancer les tests 10 fois et faites une moyenne. Utiliser `Date.now();` pour mesurer le temps.
3. Conclure en expliquant la différence de performance.

Exercice 7 : calculs en une seule passe

Le fichier de réponses pour cet exercice est nommé `MIF18-TP-MapReduce-Ex7.js`.

On souhaite calculer *cinq statistiques* sur les populations des états des États-Unis : le nombre d'états, la population totale des États-Unis, la population de l'état le plus peuplé, la moyenne des populations des états ainsi que l'écart type sur les populations des états et ceci en utilisant *un seul job Map/Reduce* sur la collection `states`. On rappelle la définition de la moyenne notée $\overline{x}$ et les deux définitions équivalentes de l'écart-type noté σ :

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \text{ et } \sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}} = \sqrt{\frac{(\sum_{i=1}^{n} x_i^2) - n\overline{x}^2}{n-1}}$$

1. Entre les deux définitions équivalentes de l'écart-type, justifier le choix de celle à utiliser.
2. Donner la fonction `exo7map`, la fonction `exo7red` et la fonction `exo7fin` permettant de calculer les statistiques avec le nombre d'états dans le champs `nb`, la population totale `sum`, la population de l'état le plus peuplé dans `max`, la moyenne dans `avg` et l'écart-type dans `std`.

6. http://docs.mongodb.org/manual/reference/command/mapReduce/#mapreduce-finalize-cmd
7. http://docs.mongodb.org/manual/core/aggregation-pipeline/
8. http://docs.mongodb.org/manual/tutorial/aggregation-zip-code-data-set/