BIG DATA - M2 INFO

2023 / 2024

Module : Big data
Enseignant : Laurent Amanton
Groupe : Hajar RAHMOUNI, Firdaous EL HALAFI, Nathan MARYE

TP requêtes avec MongoDB uniquement

1. Écrire les requêtes simples suivantes

1.1. Nombre de publications présentes dans la collection

```
use DBLP
```

```
db.createCollection("publis")
db.publis.createIndex({year:1})
sh.enableSharding("DBLP")
sh.shardCollection("DBLP.publis", {year:"hashed"})
db.publis.count()
```

1.2. Liste des livres (book) parus après 2013

```
db.publis.find({year:{$gt:2013},type:"Book"})
```

1.3. Liste des éditeurs (distinct)

```
db.publis.distinct("editor")
```

1.4. Nombre de publications par auteur

```
db.publis.aggregate([
 { $unwind: "$authors"},
 {"$group": {
      _id: "$authors",
      nb: {$sum: 1}
    }
 },
 { $sort : { nb : -1}}
]);
```

2. Map Reduce

2.1. Nombre de pages de tous les documents

```
var mapFunction = function() {
   if (this.pages && this.pages.start !== undefined && this.pages.end
!== undefined) {
       var numberOfPages = this.pages.end - this.pages.start;
       emit("total", numberOfPages);
   }
};

var reduceFunction = function(key, values) {
   return Array.sum(values);
};

var startTimeMapReduce = new Date();

var result = db.publis.mapReduce(
   mapFunction,
   reduceFunction,
   { out: { inline: 1 } }
);

var endTimeMapReduce = new Date();

var durationMapReduce = endTimeMapReduce - startTimeMapReduce;

print("Durée d'exécution de Map-Reduce : " + durationMapReduce + "
millisecondes");

print(result);
```

Sortie:

```
Durée d'exécution de Map-Reduce : 271 millisecondes
1727477
```

2.2. Nombre de pages de tous les livres (book)

```
var mapFunction = function() {
   if (this.type === "Book" && this.pages && this.pages.start !==
undefined && this.pages.end !== undefined) {
       var numberOfPages = this.pages.end - this.pages.start;
       emit("total", numberOfPages);
   }
};

var reduceFunction = function(key, values) {
```

```
   return Array.sum(values);
};

var startTimeMapReduce = new Date();

var result = db.publis.mapReduce(
   mapFunction,
   reduceFunction,
   { out: { inline: 1 } }
);

var endTimeMapReduce = new Date();

var durationMapReduce = endTimeMapReduce - startTimeMapReduce;

print("Durée d'exécution de Map-Reduce : " + durationMapReduce + "
millisecondes");

print(result);
```

Sortie:

```
Durée d'exécution de Map-Reduce : 235 millisecondes
445309
```

3. Aggregate

3.1. Nombre de pages de tous les documents

```
db.publicationsCollection.aggregate([
 {
   $group: {
     _id: null,
     totalPages: {
       $sum: {
         $subtract: ["$pages.end", "$pages.start"]
       }
     }
   }
 }
]).explain("executionStats")
```

Temps d'exécution : { shard 1: 8ms et shard 2: 77ms }

3.2. Nombre de pages de tous les livres (book)

```
db.publicationsCollection.aggregate([
 {
   $match: {
     type: "Book"
   }
 },
 {
   $group: {
     _id: null,
     totalPages: {
       $sum: {
         $subtract: ["$pages.end", "$pages.start"]
       }
     }
   }
 }
]).explain("executionStats")
```

Temps d'exécution : 23ms

3.3 Nombre de publications par auteur

```
db.publicationsCollection.aggregate([
 {
   $unwind: "$authors"
 },
 {
   $group: {
     _id: "$authors",
     count: { $sum: 1 }
   }
 }
]).explain("executionStats")
```

Temps d’exécution : { shard 1: 22ms, shard 2: 270ms }

TP sharding (MongoDB et CouchDB)

1. Test de la base distribuée

1.1. Donnez le nombre de documents par shard

- Créer la base de données DBLP:

```
use DBLP
db.createCollection("publis")
db.publis.createIndex({year:1})
```

- Activer le sharding:

```
sh.enableSharding("DBLP")
sh.shardCollection("DBLP.publis", {year:"hashed"})
```

- Importer les données:

```
mongoimport --host router:27023 --db DBLP --collection publis <
dblp.json --jsonArray
```

- Nombre de documents par shard:

```
db.publis.getShardDistribution()
```

sortie:

```
Shard PC2 at PC2/192.168.2.8:27025,192.168.2.9:27026
{
  data: '17.25MiB',
  docs: 56094,
  chunks: 1,
  'estimated data per chunk': '17.25MiB',
  'estimated docs per chunk': 56094
}
---
Shard PC1 at PC1/192.168.2.2:27018,192.168.2.3:27019,192.168.2.4:27020
{
  data: '19.39MiB',
  docs: 61932,
```

```
  chunks: 1,
  'estimated data per chunk': '19.39MiB',
  'estimated docs per chunk': 61932
}
---
Totals
{
  data: '36.65MiB',
  docs: 118026,
  chunks: 2,
  'Shard PC2': [
        '47.08 % data',
        '47.52 % docs in cluster',
        '322B avg obj size on shard'
  ],
  'Shard PC1': [
        '52.91 % data',
        '52.47 % docs in cluster',
        '328B avg obj size on shard'
  ]
}

—————————
PC2 : 56094
PC1 : 61932
```

1.2 Vérifier que l'ajout d'un nouveau document est dans le *bon* shard.

- Insertion dans PC1:

```
db.publis.insertOne({
   "_id": "series/cogtech/Zancanaro13",
   "type": "Article",
   "title": "Shared Interfaces for Co-located Interaction.",
   "pages": {"start": 71, "end": 88},
   "year": 2010,
   "booktitle": "Ubiquitous Display Environments",
   "url": "db/series/cogtech/364227662.html#Zancanaro12",
   "authors": ["Massimo Zancanaro"]
})
```

- Insertion dans PC2:

```
db.publis.insertOne({
   "_id": "series/cogtech/Zancanaro5",
   "type": "Article",
   "title": "Shared Interfaces for Co-located Interaction.",
   "pages": {"start": 71, "end": 88},
   "year": 2020,
   "booktitle": "Ubiquitous Display Environments",
   "url": "db/series/cogtech/364227662.html#Zancanaro12",
   "authors": ["Massimo Zancanaro"]
})
```

1.3 Tester la dernière requête *mapReduce* du TP précédent et comparer les vitesses d'exécution (avec et sans shard)

- Requête avec aggregate avec shard:

```
var startTime = new Date();

db.publis.aggregate([
 { $unwind: "$authors"},
 {"$group": {
      _id: "$authors",
      nb: {$sum: 1}
    }
 },
 { $sort : { nb : -1}}
]);

var endTime = new Date();

var duration = endTime - startTime;

print("Durée d'exécution de Map-Reduce : " + duration + "
millisecondes");
```

- Sortie:

```
Durée d'exécution : 751 millisecondes
```

- Requête avec map reduce avec shard:

```
var mapFunction = function() {
   if (this.authors) {
       this.authors.forEach(function(author) {
           emit(author, 1);
       });
```

```
    }
};

var reduceFunction = function(key, values) {
    return Array.sum(values);
};

var startTimeMapReduce = new Date();

var result = db.publis.mapReduce(
    mapFunction,
    reduceFunction,
    { out: { inline: 1 } }
);

var endTimeMapReduce = new Date();

var durationMapReduce = endTimeMapReduce - startTimeMapReduce;

print("Durée d'exécution de Map-Reduce : " + durationMapReduce + "
millisecondes");

print(result);
```

- Sortie

```
Durée d'exécution de Map-Reduce : 2309 millisecondes
```

- Requête avec aggregate sans shard:

```
var startTime = new Date();

db.publis.aggregate([
 { $unwind: "$authors"},
 {"$group": {
       _id: "$authors",
       nb: {$sum: 1}
     }
 },
 { $sort : { nb : -1}}
]);

var endTime = new Date();

var duration = endTime - startTime;

print("Durée d'exécution de Map-Reduce : " + duration + "
millisecondes");
```

- Sortie

```
Durée d'exécution de aggregate : 532 millisecondes
```

- Requête avec map reduce sans shard:

```
var mapFunction = function() {
   if (this.authors) {
       this.authors.forEach(function(author) {
           emit(author, 1);
       });
   }
};

var reduceFunction = function(key, values) {
   return Array.sum(values);
};

var startTimeMapReduce = new Date();

var result = db.publis.mapReduce(
   mapFunction,
   reduceFunction,
   { out: { inline: 1 } }
);

var endTimeMapReduce = new Date();

var durationMapReduce = endTimeMapReduce - startTimeMapReduce;

print("Durée d'exécution de Map-Reduce : " + durationMapReduce + "
millisecondes");

print(result);
```

- Sortie

```
Durée d'exécution de Map-Reduce : 2099 millisecondes
```

2. Incidence de la clé de partitionnement sur les performance des requêtes longues

2.1. Importer la base Personnes (200 000 personnes).

Se connecter au “routeur” de la base de données permettant d’effectuer les requêtes

```
mongosh
```

Utiliser une base de données spécifique

```
use databaseSharding
```

Créer une collection de documents de type “personne”

```
db.createCollection("personsCollection")
```

Quitter le routeur

```
exit
```

Importer le fichier “persons.json” dans MongoDB, dans la bonne base de données et dans la bonne collection.

```
mongoimport \
   --db databaseSharding \
   --collection personsCollection \
   --file persons.json \
   --jsonArray
```

2.2. Créez une requête simple (couleur des yeux) et une requête complexe sur cette base.

Requête simple

```
db.personsCollection.countDocuments({eyeColor: "brown"})
```

Requête complexe

```
db.publis.aggregate([
   { $match: { eyeColor: "green", age: { $gte: 25 } } },
   { $group: { _id: "$gender", count: { $sum: 1 } } }
])
```

2.3. Tester ces deux requêtes en modifiant la clé de répartition pour montrer la pertinence de votre choix de clé (pour chaque requête).

- avec l’index sur gender:

```
db.publis.createIndex({gender:1})
sh.enableSharding("Personnes")
sh.shardCollection("Personnes.publis", {gender:"hashed"})
```

- Pour la première requête:

```
var startTime = Date();

db.publis.find({"eyeColor": "brown"});

var endTime = Date();

var executionTime = endTime - startTime;

print("Temps d'exécution : " + executionTime + " ms");
```

- Sortie:

```
Temps d'exécution : NaN ms
```

- Pour la deuxième requête:

```
var start = new Date();

db.publis.aggregate([
   { $match: { eyeColor: "green", age: { $gte: 25 } } },
   { $group: { _id: "$gender", count: { $sum: 1 } } }
])

var end = new Date();

print("Temps d'exécution : " + (end - start) + " millisecondes");
```

- Sortie:

```
Temps d'exécution : 103 millisecondes
```

- avec l’index sur age:

```
db.publis.createIndex({age:1})
sh.enableSharding("Personnes")
sh.shardCollection("Personnes.publis", {age:"hashed"})
```

- Pour la première requête:

```
var startTime = Date();

db.publis.find({"eyeColor": "brown"});

var endTime = Date();

var executionTime = endTime - startTime;

print("Temps d'exécution : " + executionTime + " ms");
```

- Sortie:

```
Temps d'exécution : NaN ms
```

- Pour la deuxième requête:

```
var start = new Date();

db.publis.aggregate([
   { $match: { eyeColor: "green", age: { $gte: 25 } } },
   { $group: { _id: "$gender", count: { $sum: 1 } } }
])

var end = new Date();

print("Temps d'exécution : " + (end - start) + " millisecondes");
```

- Sortie:

```
Temps d'exécution : 172 millisecondes
```

3. CouchDB

3.1. Tester la dernière requête mapReduce du TP précédent et comparer les vitesses d'exécution (avec et sans shard)

- Sans shards:

```
curl -X PUT "$COUCH/publications/_design/authors" -H "Content-Type:
application/json" -d '{
 "views": {
   "publications_by_author": {
     "map": "function (doc) { doc.authors.forEach(function (author) {
emit(author, 1); }); }",
     "reduce": "_sum"
   }
 }
}'
time curl
"$COUCH/publications/_design/authors/_view/publications_by_author?gro
up=true" -o /dev/null
```

- Sortie :

Temps d'exécution : 2 064 ms

- Avec shards :

```
curl -X PUT "$COUCH/publications/_design/authors" -H "Content-Type:
application/json" -d '{
 "views": {
   "publications_by_author": {
     "map": "function (doc) { doc.authors.forEach(function (author) {
emit(author, 1); }); }",
     "reduce": "_sum"
   }
 }
}'
time curl
"$COUCH/publications/_design/authors/_view/publications_by_author?gro
up=true" -o /dev/null
```

- Sortie

Temps d'exécution : 4259 ms$

- Création de la base de données shardée avec 8 shards de 3 réplicas

````shell
curl -X PUT "http://admin:password@localhost:5984/personne?q=8&n=3"
````

- Import des données depuis le fichier Personnes.json

````shell
curl -X POST -H "Content-Type: application/json"
http://admin:password@localhost:5984/personne/_bulk_docs --data
'@chemin_vers_fichier/CouchDB/Personnes.json'
````

3.2. Créez une requête simple (couleur des yeux) et une requête complexe sur cette base.

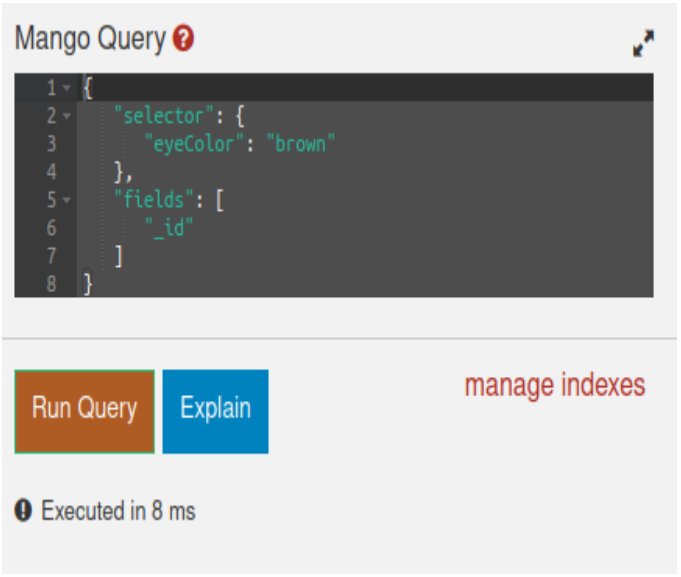
3.2.1 Requête simple

query.json

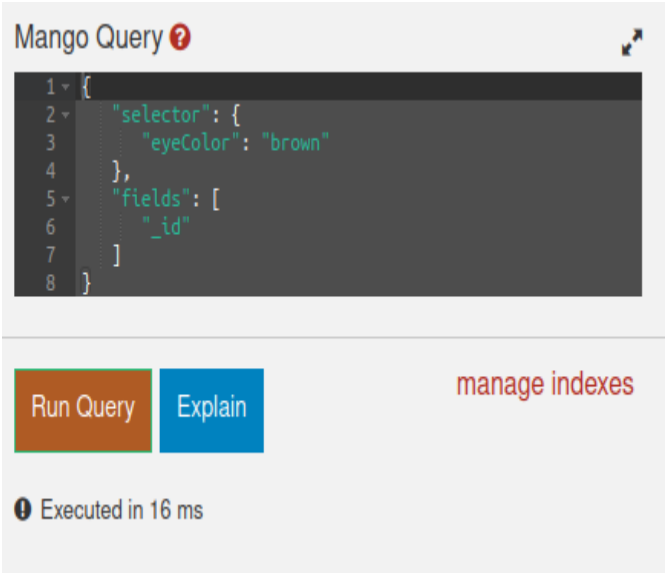
````json
{
 "selector": {
   "eyeColor": "brown"
 },
 "fields": ["_id"]
}
````

````shell
curl -X POST -H "Content-Type: application/json" \
 "http://admin:password@localhost:5984/personne/_find" \
 --data "@query.json"
````

- avec le partitionnement automatique



- avec la clé de partition : name



3.2.2 Requête complexe

query.json

````
```json
{
 "selector": {
   "$and": [
     {
       "eyeColor": "green"
     },
     {
       "age": {
         "$gte": 25
````
````

```
        }
      }
    ]
  },
  "fields": [
    "_id",
    "gender"
  ]
}
````

````shell
curl -X POST -H "Content-Type: application/json" \
 "http://admin:password@localhost:5984/personne/_find" \
 --data "@query.json"
````

- avec le partitionnement automatique de CouchDB

- avec la clé de partition : name

````shell
curl -X PUT
"http://admin:password@localhost:5984/personne2?partition_key=name&q=4&n=2"
````

Conclusion

En analysant les résultats de nos requêtes, il est apparent que MongoDB affiche généralement des performances plus élevées que CouchDB.