

Projet de BigData

Les bases NoSQL

par Jordan Baudin, Corentin LeGuen et Geoffrey Spaur

11 janvier 2018

Contents

1	Présentation	3
2	La base de donnée CouchBase	4
2.1	Les modalités d'installation	4
2.2	Présentation	6
2.3	Programme Java	7
2.4	Performances	9
3	La base de donnée MongoDB	10
3.1	Les modalités d'installation	10
3.2	Les méthodes d'insertion de données	11
3.3	Le langage de recherche	12
3.3.1	Les conditions	13
3.3.2	L'affichage	14
3.4	L'indexation interne	14
3.5	Le support de la concurrence d'accès	15
3.6	L'architecture du système	16
3.7	Les techniques de distribution	16
4	Conclusion	17

1 Présentation

Ce projet a pour but de comparer différentes bases de données NoSQL. Nous allons prendre les bases données CouchBase et MongoDB.

2 La base de donnée CouchBase

2.1 Les modalités d'installation

Recommandations système

CPU	6 coeurs 64-bits à 3GHz
RAM	16 GB physique

TODO

Installation

L'installation de CouchBase peut se faire via le site officiel avec un .dpkg soit en ligne de commande :

```
corentin@asus:~$ sudo apt-get install couch  
couchapp couchdb couchdb-common  
couchbase-server couchdb-bin
```

Le service couchbase-server utilise les ports suivants :

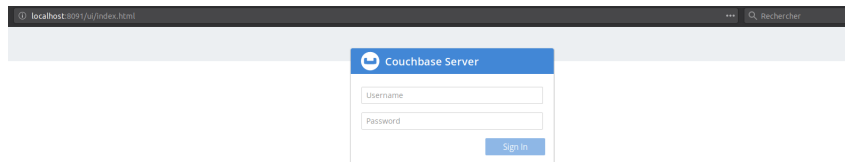
```
4369/tcp open epmc  
6060/tcp open x11  
8091/tcp open jamlink  
8092/tcp open unknown  
8093/tcp open unknown  
8094/tcp open unknown  
9100/tcp open jetdirect  
9101/tcp open jetdirect  
9102/tcp open jetdirect  
9998/tcp open distinct32  
9999/tcp open abyss  
11207/tcp open unknown  
11209/tcp open unknown  
11210/tcp open unknown  
11214/tcp open unknown  
11215/tcp open unknown  
18091/tcp open unknown  
18092/tcp open unknown  
18093/tcp open unknown  
18094/tcp open unknown  
19102/tcp open unknown  
21100/tcp open unknown  
21101/tcp open unknown  
35255/tcp open unknown
```

Le service couchbase-server peut être coupé avec la ligne de code

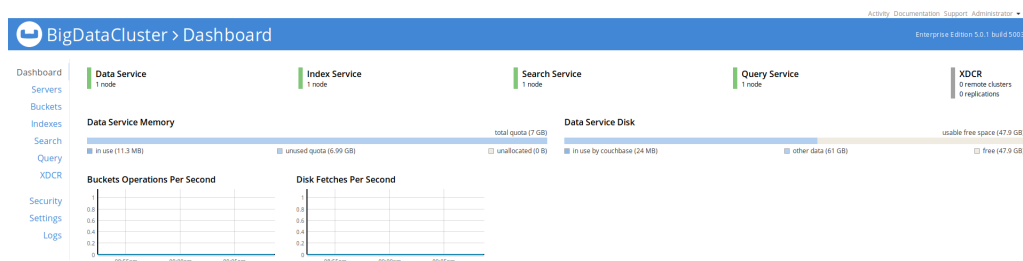
```
sudo service apt-get couchbase-server stop
```

2.2 Présentation

Le service couchbase est disponible à l'adresse localhost:8091/ui/index.html



Par défaut, lors de la première utilisation, le service demande un mot de passe pour l'utilisateur Administrator.

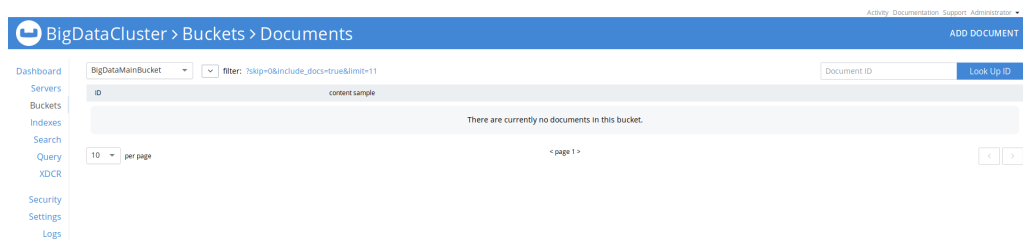


Sur le dashboard, on peut voir les statistiques d'utilisation et d'accès à CouchBase.

Avant la toute première utilisation, il est nécessaire de créer un Bucket (qui peut être vu comme un cluster) pour pouvoir insérer et manipuler des données.

name	items	resident	ops/sec	RAM used/quota	disk used	
BigDataMainBucket	0	100%	0	11.3MB / 7GB	24MB	Documents Statistics

Ici, j'ai créé un bucket BigDataMainBucket.



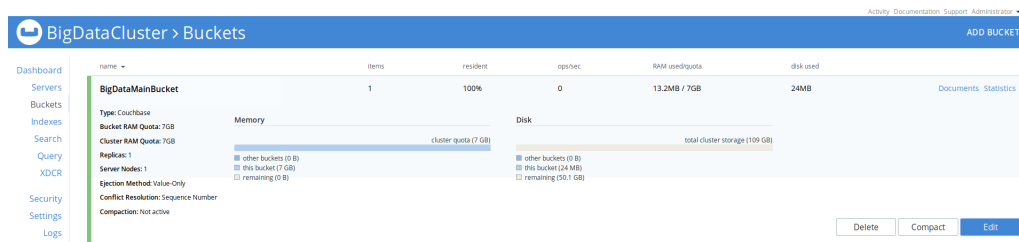
2.3 Programme Java

Nous avons développé un programme en Java pour insérer des données dans mon bucket. Le code source est disponible ici.

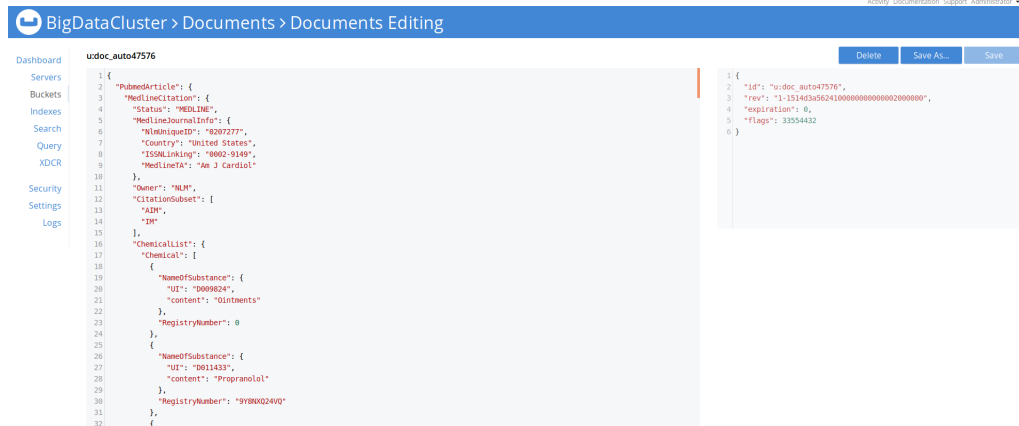
Une documentation complète et un code riche et détaillé y est présente. Le programme repose sur un JDBC CouchBase et sur un parseurJSON fait en java :

```
public static void parseJson(JSONObject jsonObj) throws ParseException {  
    Set<Object> set = jsonObj.keySet();  
    Iterator<Object> iterator = set.iterator();  
    while (iterator.hasNext()) {  
        Object obj = iterator.next();  
        if (jsonObj.get(obj) instanceof JSONArray) {  
            System.out.println(obj.toString());  
            getArray(jsonObj.get(obj));  
        } else {  
            if (jsonObj.get(obj) instanceof JSONObject) {  
                parseJson((JSONObject) jsonObj.get(obj));  
            } else {  
                System.out.println(obj.toString() + "\t"  
                    + jsonObj.get(obj));  
            }  
        }  
    }  
}
```

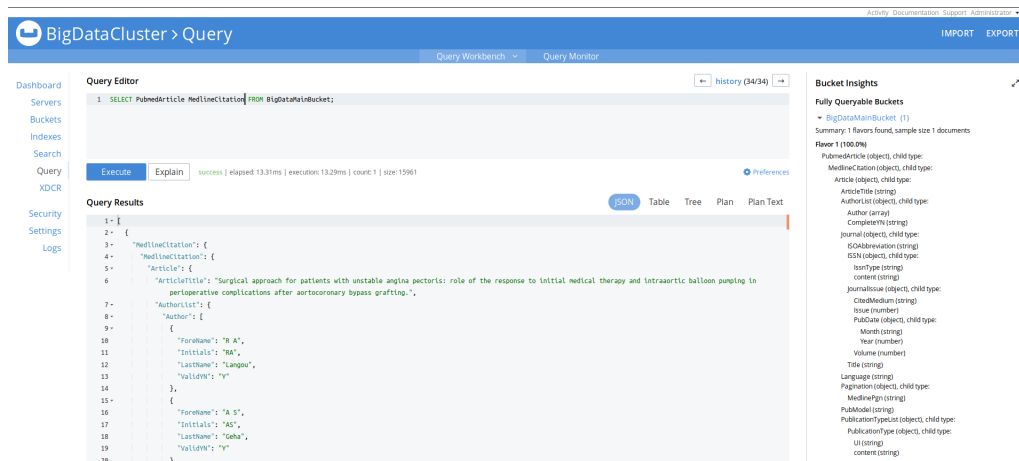
Après l'insertion d'un élément en JSON, grâce à l'AJAX, le résultat est affiché directement (pas besoin de recharger la page).



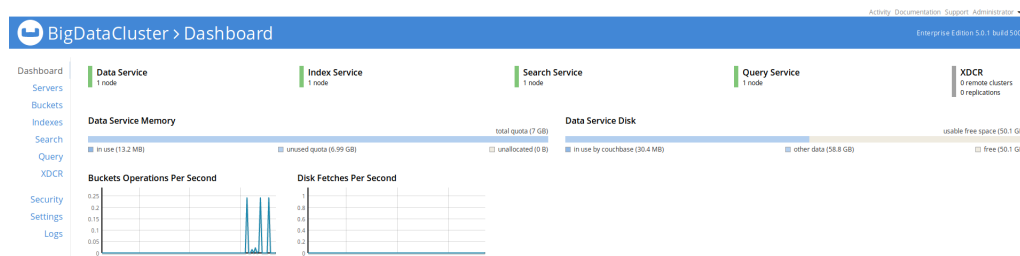
CouchDB permet de visionner directement les documents enregistrés.



On peut également réaliser des requêtes NIQL sur CouchDB.



Le dashboard permet de visionner les opérations réalisées sur le serveur.



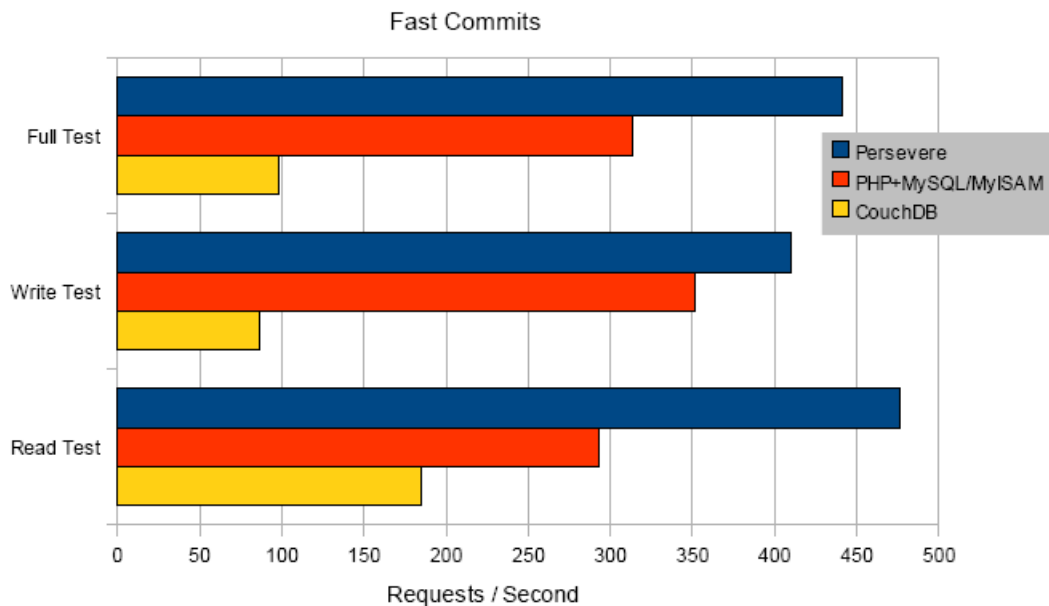
2.4 Performances

Les performances de CouchDB sont disponibles sur le site d'arangoDB

Here are sequential bulk document inserts at four different granularities, from an array of 100 documents, up through 1,000, 5,000, and 10,000:

bulk_doc_100	4400 docs	437.37574552683895 docs/sec
bulk_doc_1000	17000 docs	1635.4016354016355 docs/sec
bulk_doc_5000	30000 docs	2508.1514923501377 docs/sec
bulk_doc_10000	30000 docs	2699.541078016737 docs/sec

Voici les performances comparé à Persevere et PHP+Mysql :



3 La base de donnée MongoDB

3.1 Les modalités d'installation

Recommandations système Il est recommandé par la documentation d'utiliser des processeurs 64-bits car sinon, la mémoire sera limitée à 2 GB. Les performances du serveur sont fortement dépendantes de la RAM du système.

Téléchargement Pour l'installation de MongoDB, il vous suffira dans un premier temps de télécharger l'archive contenant MongoDB.

Après le téléchargement de l'archive, vous pourrez la décompresser. Avant de lancer le serveur, il vous faudra créer le dossier **/data/db**. Ce dossier contiendra toutes les bases de MongoDB.

Enfin vous pouvez lancer le serveur MongoDB avec la commande:

```
$ ./bin/mongod
```

```
geoffrey@Debian:~/git/s9/BD/mongodb-linux-x86_64-debian81-3.6.1$ ./bin/mongod
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] MongoDB starting : pid=2409
port=27017 dbpath=/data/db 64-bit host=Debian
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] db version v3.6.1
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] git version: 025d4f4fe61efd1fb6f0005be20cb45a004093d1
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1t 3 May 2016
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] allocator: tcmalloc
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] modules: none
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] build environment:
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] distmod: debian81
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] distarch: x86_64
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] target_arch: x86_64
2018-01-11T14:28:55.444+0100 I CONTROL [initandlisten] options: {}
2018-01-11T14:28:55.508+0100 I - [initandlisten] Detected data files in /data/db created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2018-01-11T14:28:55.508+0100 I STORAGE [initandlisten]
2018-01-11T14:28:55.508+0100 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2018-01-11T14:28:55.508+0100 I STORAGE [initandlisten] ** See http://docs.mongodb.org/core/prodnotes-filesystem
2018-01-11T14:28:55.524+0100 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=1427M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
2018-01-11T14:28:56.039+0100 I STORAGE [initandlisten] WiredTiger message [1515677336:39864][2409:0x7fc29f566a00], txn-recover: Main recovery loop: starting at 2/10752
2018-01-11T14:28:56.152+0100 I STORAGE [initandlisten] WiredTiger message [1515677336:152674][2409:0x7fc29f566a00], txn-recover: Recovering log 2 through 3
2018-01-11T14:28:56.269+0100 I STORAGE [initandlisten] WiredTiger message [1515677336:269196][2409:0x7fc29f566a00], txn-recover: Recovering log 3 through 3
2018-01-11T14:28:57.010+0100 I CONTROL [initandlisten]
2018-01-11T14:28:57.010+0100 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-01-11T14:28:57.010+0100 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2018-01-11T14:28:57.010+0100 I CONTROL [initandlisten]
2018-01-11T14:28:57.010+0100 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2018-01-11T14:28:57.010+0100 I CONTROL [initandlisten] ** Remote systems
```

Vous pourrez ensuite accéder à MongoDB en lançant le client grâce à la commande suivante:

```
$ ./bin/mongo
```

3.2 Les méthodes d'insertion de données

Création de la base Avant toute insertion, lancez le client MongoDB avec la commande précédente. Nous allons ajouter une nouvelle base de donnée. Vous pouvez lister les différentes bases de données déjà présentes avec la commande:

```
> show dbs
```

```
> show dbs
admin      0.000GB
bigdata    0.000GB
config     0.000GB
local      0.000GB
```

Puis pour créer et/ou utiliser notre base de donnée, utilisez:

```
> use DATABASE
```

```
> use bigdata
switched to db bigdata
```

Vous pouvez lister vos collections de document avec la commande suivante:

```
> show collections
```

```
> show collections
alcoholism
articles
medarticles
```

Insertion de documents Vous pouvez maintenant sortir du client MongoDB. Vos documents doivent être en format json sous forme d'objet. Vous trouverez un exemple avec le fichier *prettyAlcoholism.json*. Nous allons utiliser la commande *mongoimport* afin d'ajouter nos document dans notre base:

```
$ ./bin/mongoimport --db bigdata --collection medarticles
--file ../Projet/prettyAlcoholism.json
```

```
geoffrey@Debian:~/git/s9/BD/mongodb-linux-x86_64-debian81-3.6.1$ ./bin/mongoimport
--db bigdata --collection medarticles --file ../Projet/prettyAlcoholism.json
2018-01-11T15:31:59.658+0100    connected to: localhost
2018-01-11T15:31:59.858+0100    imported 20 documents
```

Vous pouvez cependant ajouter des document directement à partir du client MongoDB avec les commandes *import* ou *importMany*. Ces commandes sont utiles dans le cas où nous avons peu de document ou des document de petites tailles.

3.3 Le langage de recherche

Pour rechercher des documents dans MongoDB, nous utiliserons la fonction *find()* sur nos collections. Cette commande peut prendre jusqu'à deux arguments sous format JSON:

- Le premier permet de déterminer la clause **where**.
- La seconde permet de spécifier les champs à afficher.

3.3.1 Les conditions

Vous pouvez effectuer une recherche en indiquant la valeur d'un champs comme ceci:

```
> db.medarticles.find({"MedlineCitation.PMID.content":29054077}).pretty()
{
  "_id" : ObjectId("5a577331ca1952a3c0263e3a"),
  "MedlineCitation" : {
    "Status" : "Publisher",
    "Owner" : "NLM",
    "PMID" : {
      "Version" : 1,
      "content" : 29054077
    },
    "DateCreated" : {
      "Year" : 2017,
      "Month" : 10,
      "Day" : 20
    },
    "DateRevised" : {
      "Year" : 2017,
      "Month" : 10,
      "Day" : 20
    },
    "Article" : {
      "PubModel" : "Print-Electronic",
      "Journal" : {
        "ISSN" : {
          "IssnType" : "Electronic",
          "content" : "1873-6327"
        },
        "JournalIssue" : {
          "CitedMedium" : "Internet",
          "Volume" : 77,
          "PubDate" : {
            "Year" : 2017,
            "Month" : "Oct",
            "Day" : 3
          }
        },
        "Title" : "Addictive behaviors",
        "ISOAbbreviation" : "Addict Behav"
      },
      "ArticleTitle" : "Relationship between empathic processing
and drinking behavior in project MATCH.",
      "Pagination" : {
        "MedlinePgn" : "180-186"
      }
    }
  }
}
```

En ajoutant la fonction *pretty()*, le résultat sera indenté. Vous pouvez utiliser les tableaux *\$and* et *\$or* afin de constituer des recherches plus complexes.

3.3.2 L'affichage

Vous pouvez aussi n'afficher que certain champs dans le résultat de votre recherche:

```
> db.medarticles.find({"MedlineCitation.PMID.content":29054077}, {"MedlineCitation.Article.ArticleTitle":1}).pretty()
{
  "_id" : ObjectId("5a57755fca1952a3c0263e69"),
  "MedlineCitation" : {
    "Article" : {
      "ArticleTitle" : "Relationship between empathic processing
and drinking behavior in project MATCH."
    }
  }
}
```

3.4 L'indexation interne

Pour chaque document ajouter dans une collection, MongoDB lui ajoute un champs `_id`. C'est ce champs qui sera indexé par défaut. Il est possible de lister tous les indexes d'une collection avec la commande suivant:

```
> db.medarticles.getIndexes()
```

```
> db.medarticles.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "bigdata.medarticles"
  }
]
```

Vous pouvez cependant ajouter vos propres indexes avec la commande `ensureIndex()`. L'objet JSON passé en argument doit être une liste de `int`. Le nom doit être le champs à indexer et le `int` doit être égal à 1 ou -1; la valeur 1 correspond à un tri ascendant et la valeur -1 à un tri descendant.

```
> db.medarticles.ensureIndex({"MedlineCitation.PMID.content":1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
```

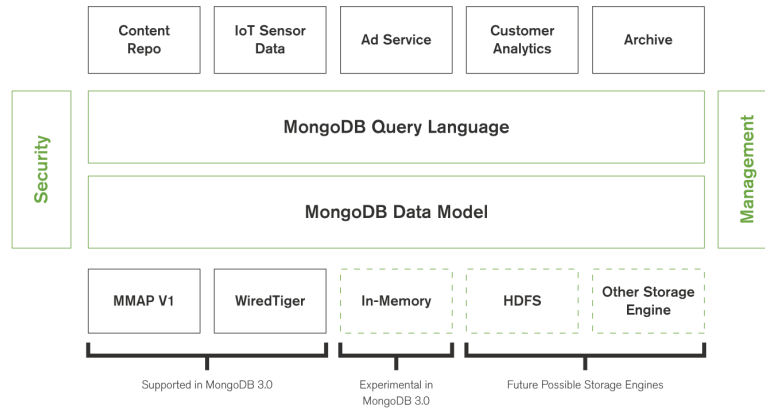
Afin de mesurer l'impact de vos indexes, à chaque recherche vous pouvez ajouter la méthode `explain`:

```
> db.medarticles.find({"MedlineCitation.PMID.content":29054077}).explain()
{
  "queryPlanner" : {
    "plannerVersion" : 1,
    "namespace" : "bigdata.medarticles",
    "indexFilterSet" : false,
    "parsedQuery" : {
      "MedlineCitation.PMID.content" : {
        "$eq" : 29054077
      }
    },
    "winningPlan" : {
      "stage" : "FETCH",
      "inputStage" : {
        "stage" : "IXSCAN",
        "keyPattern" : {
          "MedlineCitation.PMID.content" : 1
        },
        "indexName" : "MedlineCitation.PMID.content_1",
        "isMultiKey" : false,
        "multiKeyPaths" : {
          "MedlineCitation.PMID.content" : [ ]
        },
        "isUnique" : false,
        "isSparse" : false,
        "isPartial" : false,
        "indexVersion" : 2,
        "direction" : "forward",
        "indexBounds" : {
          "MedlineCitation.PMID.content" : [
            "[29054077.0, 29054077.0]"
          ]
        }
      },
      "rejectedPlans" : [ ]
    },
    "serverInfo" : {
      "host" : "Debian",
      "port" : 27017,
      "version" : "3.6.1",
      "gitVersion" : "025d4f4fe61efd1fb6f0005be20cb45a004093d1"
    },
    "ok" : 1
  }
}
```

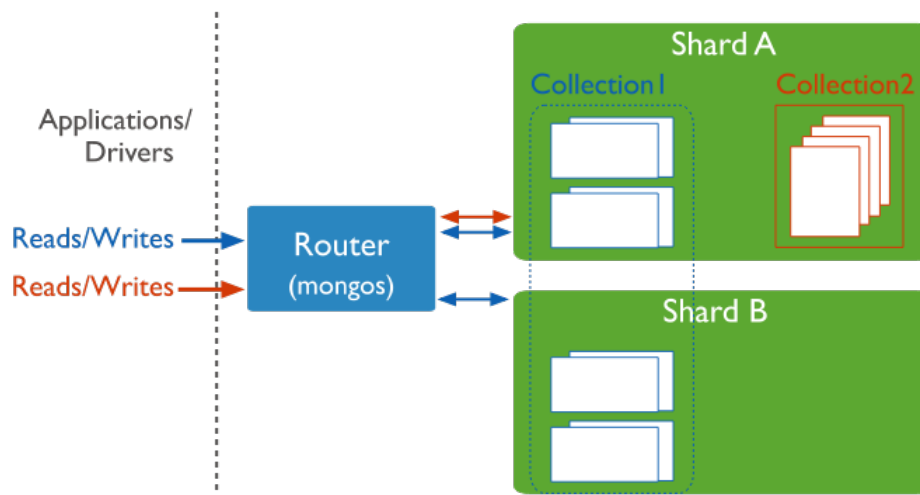
3.5 Le support de la concurrence d'accès

MongoDB support totalement la concurrence d'accès. Il permet à plusieurs clients de lire et écrire les mêmes données. MongoDB gère la concurrence à l'aide de verrous. Certaines opérations entraînent un verrouillage de la collection ou même de la base de données. Par exemple les opérations d'insertion, de suppression, de modification ou encore de création d'index entraînent un verrouillage de la base de données.

3.6 L'architecture du système



3.7 Les techniques de distribution



4 Conclusion

CouchDB et MongoDB sont des bases de données avec des objectifs différents, MongoDB sera utilisé pour interagir avec une application liée à un JDBC alors que CouchDB sera plus largement utilisé comme une interface REST, grâce à sa construction basée sur du HTTP.

MongoDB est plus rapide que CouchDB, mais ne permet pas son utilisation sur téléphone. Une base MongoDB pourra être agrandie avec facilité là où CouchDB aura plus de difficulté, mais CouchDB reste le meilleur choix pour une utilisation mobile, avec une réplication maître à maître ou un unique serveur.