TP4 couchDB
Réalisé par Amal TAOUS
Révisé par Mr SAMIR Youcef

Introduction à CouchDB

Qu’est-ce que CouchDB ?

Dans le cadre de ce TP, nous allons explorer CouchDB, une base de données NoSQL qui stocke les informations sous forme de documents JSON. Ces documents peuvent être organisés et manipulés grâce à des vues utilisant le paradigme MapReduce. CouchDB est conçu pour offrir une grande flexibilité et permettre le traitement efficace de volumes importants de données dans un environnement distribué.

Pour ce TP, nous utiliserons CouchDB afin de stocker des matrices représentant les connexions entre différentes pages web. Nous exploiterons également les capacités de traitement MapReduce pour calculer des résultats basés sur ces données.

Caractéristiques principales de CouchDB :

1. **Simplicité** : Facile à installer et à utiliser.
2. **Open Source** : Distribué sous licence Apache.
3. **API REST intuitive** :
    a. **GET** : Permet de récupérer des ressources.
    b. **PUT** : Utilisé pour créer de nouvelles ressources.

c. **POST** : Permet d'envoyer des données.

d. **DELETE** : Sert à supprimer des ressources.

À travers ce TP, nous allons mettre en œuvre ces concepts pour mieux comprendre le fonctionnement de CouchDB et son utilisation dans des scénarios pratiques.

1.2 Installation de CouchDB

Via Docker

Pour exécuter CouchDB avec Docker, utilise la commande suivante :

```
 docker run -d -- name couchDbDemo -e
COUCHDB_USER = abc -e COUCHDB_PASSWORD
=123 -p 5984:5984 couchdb
```

1.3 Vérification du serveur CouchDB

Pour vérifier si CouchDB est actif :

```
curl -X GET http :// abc :123 @localhost :5984/
```

2 Manipulation des bases de données

2.1 Création d'une base de données

Pour créer une base de données appelée films :

```
curl -X PUT http :// abc :123 @localhost :5984/ films
```

2.2 Insertion de documents Insertion d'un document simple

```
curl -X PUT http :// abc :123 @localhost :5984/ films /
doc1 -d '{" titre ":" Inception ", " annee ":2010} '
```

Insertion en masse Pour insérer plusieurs documents :

```
 curl -X POST http :// abc :123 @localhost :5984/ films
/ _bulk_docs -d @films_couchdb . json -H " Content -
Type : application / json "
```

Exemple de fichier JSON :

```
{  " docs ": [
      {" _id ": " movie :1001" , " titre ": " Inception " , "
annee ": 2010} ,
      {" _id ": " movie :1002" , " titre ": " Interstellar " , "
annee ": 2014}
      ]
}
```

2.3 Récupération d'un document

Pour récupérer un document avec un identifiant donné :

```
curl -X GET http :// abc :123 @localhost :5984/ films /
movie :1001
```

3. MapReduce avec CouchDB

3.1 Qu’est-ce que MapReduce ?

MapReduce est une méthode de traitement des données conçue pour gérer de très grandes quantités d'informations, souvent réparties sur plusieurs machines. Elle se décompose en deux étapes simples :

a. **Map** : Chaque morceau de données est analysé séparément pour produire des paires clé-valeur (par exemple, une clé peut être un mot, et la valeur son nombre d'apparitions).
b. **Reduce** : Les paires ayant la même clé sont regroupées et leurs valeurs combinées pour obtenir un résultat global.

Cela permet de diviser le travail, de le répartir efficacement , et ainsi de traiter de grands volumes de données plus rapidement et de manière organisée.

3.2 Exemple : Nombre de films par année

3.2.1 Fonction Map

La fonction Map donne :

l’année = clé

le titre = valeur :

```
function ( doc ) {
        emit ( doc . annee , doc . titre );
 }
```

Exemple de résultat intermédiaire :

{" key ":2010 , " value ":" Inception "}

{" key ":2014 , " value ":" Interstellar "}

3.2.2 Fonction Reduce

La fonction Reduce compte le nombre de titres par année :

La fonction Reduce compte le nombre de titres par année :

```
function ( keys , values ) {
 return values . length ; 3
}
```

Résultat :

{" key ":2010 , " value ":1}

{" key ":2014 , " value ":1}

3.3 Exemple : Nombre de films pour chaque acteur

Fonction Map

Compter le nombre de films pour chaque acteur :

```
function ( doc ) {
for ( i = 0; i < doc . actors . length ; i ++) {
emit ({" p r n o m ": doc . actors [i ]. rst_name , " nom
": doc . actors [i ]. last_name } , doc . title );
}
}
```

Fonction Reduce

Compter le nombre de films par acteur :

```
function ( keys , values ) {
return values . length ; 3
}
```

4 Conclusion CouchDB,

CouchDB est le choix parfait pour travailler avec des bases de données documentaires et traiter de grandes quantités de données.

5 Exercice n°1 : Modèle et traitement MapReduce

Exercice n° 1 : soit une matrice M de dimension $N \times N$ représentant des liens d'un très grand nombre de pages web (soit N). Chaque lien est étiqueté par un poids (son importance).

1. Proposer un modèle, sous forme de ducuments structurés, pour représenter une telle matrice (s'inspirer du cas Page Rank du moteur de recherche Google, vu en cour). Soit C la collection ainsi obtenue.
2. La ligne i peut être vue comme un vecteur à N dimensions décrivant la page P_i. Spécifiez le traitement MapReduce qui calcule la norme de ces vecteurs à partir des documents de la collection C. La norme d'un vecteur $V(v_1, v_2, \ldots, v_N)$ est le scalaire $||V|| = \sqrt{v_1^2 + v_2^2 + \ldots + v_N^2}$
3. Nous voulons calculer le produit de la matrice M avec un vecteur de dimension N, $W(w_1, w_2, \ldots, w_N)$. Le résultat est un vecteur $\phi = \sum_{j=1}^{N} M_{ij} w_j$, On suppose que le vecteur W tient en mémoire RAM et est accessible comme variable statique par toutes les fonctions de Map ou de Reduce. Spécifiez le traitement MapReduce qui implante ce calcul.

5.1 Modèle de données

Les interactions entre les pages web peuvent être représentées sous forme d'une matrice M, où chaque élément reflète la relation entre deux pages.
Dans cette structure, une ligne de la matrice correspond à une page web spécifique et décrit l'ensemble de ses connexions avec les autres pages du réseau.
Plus précisément, chaque valeur Mij traduit l'importance du lien existant entre une page Pi et une page Pj.
Cette organisation matricielle permet de capturer efficacement la nature et l'intensité des relations entre les différentes pages du réseau.

Pour une implémentation dans CouchDB, il serait judicieux de structurer chaque document de manière à représenter une page individuelle, accompagnée de l'ensemble de ses liens sortants.

```
{
       "_id": "page_1",
      "links": {
            "page_1": 0.1,
           "page_2": 0.3,
            "page_3": 0.5
      }
 }
```

Ce document représente la page P1, avec des liens vers P1, P2, et P3, et leurs poids respectifs.

5.2.1 Étape Map

```
function map(doc) {
       var sum = 0;
      for (var page in doc.links) {
          sum += Math.pow(doc.links[page], 2);
       }
      emit(doc._id, sum);
}
```

5.2.2 Étape Reduce

```
function reduce(keys, values, rereduce) {
      var total = 0;
      for (var i = 0; i < values.length; i++) {
            total += values[i];
       }
      return Math.sqrt(total);
 }
```

5.3.1 Étape Map

La fonction Map multiplie chaque poids de la matrice par l'élément correspondant du vecteur W.

```
function map(doc) {
      var vectorW ;
       var result = 0;
       for (var page in doc.links) {
           var index = parseInt(page.split('_')[1]);
           result += doc.links[page] * vectorW[index - 1];
      }
      emit(doc._id, result);
 }
```

5.3.2 Étape Reduce

```
function reduce(keys, values, rereduce) {
        var total = 0;
       for (var i = 0; i < values.length; i++) {
             total += values[i];
        }
       return total;
}
```