Rendu TP 4

Introduction

Ce rapport a pour objectif d'introduire le concept de MapReduce en mode centralisé à travers CouchDB. Nous allons d'abord expliquer ce paradigme, présenter CouchDB et ses spécificités, puis proposer une modélisation des données pour représenter une matrice de liens entre pages web. Enfin, nous spécifierons deux traitements MapReduce : l'un pour le calcul de la norme des vecteurs ligne et l'autre pour le produit d'une matrice avec un vecteur.

1. Introduction à CouchDB

CouchDB est une base de données NoSQL orientée documents qui stocke les données sous forme de documents JSON. Contrairement aux bases relationnelles classiques, elle ne repose pas sur un schéma fixe et utilise des vues pour interroger les données. Ces vues sont générées via des fonctions MapReduce, qui permettent d'effectuer des traitements efficaces directement sur les documents stockés.

Les principales caractéristiques de CouchDB incluent :

Une API RESTful pour l'interaction avec la base.

Une réplication et synchronisation efficace entre bases de données.

Une architecture orientée vers la scalabilité et la tolérance aux pannes.

2. Installation et lancement de CouchDB avec Docker

Pour installer et exécuter un serveur CouchDB en local avec Docker, suivez les étapes suivantes :

Installation de Docker (si ce n'est pas déjà fait) :

Téléchargez et installez Docker depuis https://www.docker.com/.

Exécution de CouchDB en conteneur Docker :

Exécutez la commande suivante pour télécharger et lancer CouchDB :
`docker run -d --name couchdb -e COUCHDB_USER=admin -e COUCHDB_PASSWORD=admin -p 5984:5984 couchdb`

Cette commande démarre un conteneur CouchDB en mode détaché, accessible sur le port 5984.

Le nom du compte et le mot de passe sont **admin**

Vérification du fonctionnement :

Ouvrez un navigateur et accédez à `http://localhost:5984/_utils/`.

Vous devriez voir l'interface d'administration Fauxton de CouchDB.

Arrêt et suppression du conteneur (si nécessaire) :

Pour arrêter CouchDB : `docker stop couchdb`

Pour supprimer le conteneur : `docker rm couchdb`

3. Manipulation des documents avec l'API REST

Une fois CouchDB installé, on peut interagir avec la base de données en utilisant curl.

Création d'une base de données
curl -X PUT http://admin:admin@localhost:5984/films

Explication :

-X PUT : Utilise la méthode HTTP PUT pour créer une ressource.

http://admin:admin@localhost:5984/films : Spécifie l'URL de la nouvelle base de données appelée films.

Ajout d'un document

curl -X PUT http://admin:admin@localhost:5984/films/doc1 -H "Content-Type: application/json" -d '{"titre": "Inception", "année": 2010, "genre": "Science-fiction"}'

Explication :

-H "Content-Type: application/json" : Définit le format des données envoyées.
-d : Contient les données JSON du document.
"films/doc1" : Identifiant unique du document dans la base films.

Consultation d'un document

curl -X GET http://admin:admin@localhost:5984/films/doc1
Renvoie le contenu du document doc1.

Modification d'un document

curl -X PUT http://admin:admin@localhost:5984/films/doc1 -H "Content-Type: application/json" -d '{"_rev": "1-xxxx", "titre": "Inception", "année": 2010, "genre": "Science-fiction", "réalisateur": "Christopher Nolan"}'

Explication :

"_rev" : Indique la révision actuelle du document (obtenue via GET).
Ajoute une nouvelle propriété "réalisateur".

Suppression d'un document

curl -X DELETE http://admin:admin@localhost:5984/films/doc1?rev=1-xxxx

Explication :

DELETE : Supprime le document.rev=1-xxxx : Indique la révision du document à supprimer.

Insertion d'une collection de documents

On peut insérer plusieurs documents en une seule requête en utilisant _bulk_docs :
curl -X POST http://admin:admin@localhost:5984/films/_bulk_docs -H "Content-Type: application/json" -d @films.json

Explication :

@films.json : Fichier contenant une collection de films au format JSON.

4. Introduction à MapReduce avec CouchDB

MapReduce est un paradigme de programmation permettant le traitement de grandes quantités de données en parallèle. CouchDB intègre un moteur MapReduce pour créer des vues et effectuer des agrégations sur les documents JSON.

Principe de MapReduce

MapReduce repose sur deux fonctions principales :

- **Map** : Transforme chaque document en une paire (clé, valeur).
- **Reduce** : Agrège les valeurs associées à une même clé.

Définition d'une fonction Map dans CouchDB

Une fonction Map en CouchDB est une fonction JavaScript qui émet une paire clé-valeur pour chaque document traité.

Exemple 1 : Nombre de films par année

```
function(doc) { if (doc.année) { emit(doc.année, 1); } }
```

Explication :

- Vérifie si le document possède un champ année.
- Émet une paire (année, 1).

Exécution de la fonction Map

Dans Fauxton, créer une nouvelle vue et entrer la fonction ci-dessus dans la section **Map Function**.

Exemple 2 : Nombre de films par acteur

```
function(doc) {
  if (doc.acteurs) {
    for (var i = 0; i < doc.acteurs.length; i++) {
      emit(doc.acteurs[i], 1);
    }
  }}
```

Explication :
Parcourt la liste des acteurs du film.
Émet une paire (nom de l'acteur, 1) pour chaque acteur.

Définition d'une fonction Reduce

La fonction Reduce regroupe les valeurs en fonction de leur clé et effectue une opération d'agrégation (comme une somme, une moyenne, etc.).

Exemple : Calcul du nombre de films par année

```
function(keys, values, rereduce) { return sum(values); }
```

Explication :

- values contient les valeurs associées aux clés générées par Map.
- sum(values) additionne ces valeurs pour obtenir le total par année.

Activation de Reduce

Dans Fauxton, cocher l'option **Reduce** et exécuter la vue.

Résolution de l'exercice : Représentation d'une Matrice et Calculs avec MapReduce

Modélisation de la Matrice sous forme de Documents

La matrice M représente les liens entre différentes pages web, chaque lien étant pondéré par un poids. Pour représenter cette matrice dans **CouchDB**, nous allons utiliser une collection de documents JSON où chaque document représente une ligne de la matrice.

Structure d'un Document

Chaque document dans la collection C représente une ligne i de la matrice M et contient :

- **id** : Identifiant unique de la ligne (page web Pi).
- **liens** : Liste des pages pointées depuis Pi avec leurs poids.

```
{
  "_id": "P1",
  "liens": [
    {"page": "P2", "poids": 0.3},
    {"page": "P3", "poids": 0.7}
  ]
}
```

Dans cet exemple, la page **P1** a des liens vers **P2** (poids 0.3) et **P3** (poids 0.7).
Une matrice complète serait représentée par plusieurs documents de ce type, chacun correspondant à une ligne i.

Calcul de la Norme des Vecteurs avec MapReduce

La norme d'un vecteur ligne V=(v1,v2,...,vN) est définie par :
$||V|| = \sqrt{v1^2 + v2^2 + ... + vN^2}$
Nous allons utiliser **MapReduce** pour calculer cette norme.

Fonction Map

La fonction Map extrait les poids des liens et émet leur carré comme valeur.

```
function(doc) {
  var somme = 0;
  for (var i = 0; i < doc.liens.length; i++) {
    somme += Math.pow(doc.liens[i].poids, 2);
  }
  emit(doc._id, somme);
}
```

Explication :

- Parcourt la liste des liens d'une page.
- Calcule le carré du poids de chaque lien.
- Émet **(id de la page, somme des carrés des poids)**.

Fonction Reduce

On utilise la somme des valeurs et applique une racine carrée pour obtenir la norme.

```
function(keys, values, rereduce) {
  return Math.sqrt(sum(values));
}
```

Explication :

- sum(values) additionne les carrés des poids.
- Math.sqrt() applique la racine carrée.
- Le résultat est la **norme de la ligne Vi**.

Produit Matrice-Vecteur avec MapReduce

Nous voulons calculer :
φi=j=∑Mij*Wj
où W est un vecteur en mémoire.

Fonction Map

La fonction Map multiplie chaque poids Mij par l'élément Wj correspondant.

```
function(doc) {
  var somme = 0;
  for (var i = 0; i < doc.liens.length; i++) {
    somme += doc.liens[i].poids * W[doc.liens[i].page];
  }
  emit(doc._id, somme);
}
```

Explication :

- Accède au **vecteur W** stocké en mémoire.
- Multiplie chaque poids Mij par **Wj**.
- Émet **(id de la page, somme des produits)**.