TP4 - MapReduce & CouchDB

Exercice 1

Soit une matrice M de dimension $N \times N$ représentant les liens d'un très grand nombre de pages web (soit N). Chaque lien est étiqueté par un poids (son importance).

1. Modèle sous forme de documents structurés

Le modèle doit représenter une matrice de liens entre les pages web, inspiré de l'algorithme **PageRank** de Google.

Un format structuré possible est le suivant :

- Chaque page P_i est représentée par un document.
- Ce document contient la liste des pages vers lesquelles P_i pointe, avec les poids associés.

Exemple de document JSON :

```json
{
  "page_id": "P1",
  "outlinks": [
    {"target": "P2", "weight": 0.5},
    {"target": "P3", "weight": 0.3},
    {"target": "P4", "weight": 0.2}
  ]
}
```

La collection **C** contiendra ainsi un ensemble de documents décrivant le graphe du web.

---

2. Calcul de la norme d'un vecteur avec MapReduce

La norme d'un vecteur $V(v_1, v_2, ..., v_N)$ est donnée par :

$$||V|| = \sqrt{v_1^2 + v_2^2 + \cdots + v_N^2}$$

Implémentation avec MapReduce

1. **Map** : Chaque élément v_i du vecteur est élevé au carré et émis sous la forme d'une paire clé-valeur : (clé = "somme_partielle", valeur = v_i^2)

2. **Reduce** : La somme des valeurs reçues est calculée, puis la racine carrée est appliquée.

Pseudocode :

```python
# Mapper
def mapper(key, value):
    for v in value:
        emit("somme_partielle", v**2)

# Reducer
def reducer(key, values):
    somme = sum(values)
    emit("norme", sqrt(somme))
```

---

3. Produit Matrice-Vecteur avec MapReduce

Le produit de la matrice M avec le vecteur W est :

$$\phi_i = \sum_{j=1}^{N} M_{ij} w_j$$

Implémentation avec MapReduce

1. **Map** : Pour chaque ligne i de M, multiplier chaque élément M_{ij} par w_j et émettre : (clé = i, valeur = $M_{ij} * w_j$)

2. **Reduce** : Pour chaque ligne i, sommer les valeurs reçues pour obtenir ϕ_i.

Pseudocode :

```python
# Mapper
def mapper(i, row):
    for j, M_ij in enumerate(row):
        emit(i, M_ij * W[j])

# Reducer
def reducer(i, values):
    phi_i = sum(values)
    emit(i, phi_i)
```

Cette approche permet un traitement parallèle efficace pour de grandes matrices $N \times N$.

Tutoriel1 CouchDB

Présentation de CouchDB

CouchDB est un système de gestion de bases de données NoSQL qui stocke les données sous forme de documents JSON. Il est conçu pour être distribué, tolérant aux pannes et évolutif. CouchDB est basé sur le modèle **document-store** et utilise un format de données flexible appelé **BSON** (Binary JSON).Il expose ses fonctionnalités sous forme d'une API REST. Pour rappel, une API REST repose principalement sur quatre verbes du protocole HTTP :

- **GET** : permet de demander une ressource ou sa représentation.
- **PUT** : permet de créer une nouvelle ressource ou de la remplacer si elle

existe déjà.

- **POST** : permet d'envoyer des données à une ressource (création de données sans écraser une ressource existante).
- **DELETE** : permet de supprimer une ressource.

Installation de CouchDB

Vous avez deux possibilités pour installer CouchDB :

1. **Installation en dur** : comme dans mon cas, avec Apache CouchDB, un outil open-source soutenu par la fondation Apache.
2. **Utilisation avec Docker** : attention à bien mapper les volumes pour éviter la perte de données si le conteneur est arrêté ou supprimé.

Si vous utilisez Docker, voici la commande pour lancer un conteneur CouchDB :

```sh
docker run -d --name couchdb_demo -e COUCHDB_USER=aloyse -e
COUCHDB_PASSWORD=thebest -p 5984:5984 couchdb
```

- `-d` : exécute le conteneur en arrière-plan.
- `--name couchdb_demo` : nom du conteneur.
- `-e COUCHDB_USER=aloyse -e COUCHDB_PASSWORD=thebest` : définit les variables d'environnement pour l'authentification.
- `-p 5984:5984` : mappe le port par défaut de CouchDB.

Si c'est votre première installation, l'image sera téléchargée depuis Docker Hub.

Accès à l'interface graphique

CouchDB propose une interface graphique accessible à l'adresse :

```
http://localhost:5984/_utils
```

Vous devez entrer les identifiants définis lors de l'installation (ex. : `aloyse / thebest`).

Communication avec CouchDB via `curl`

Vérification du bon fonctionnement de CouchDB

```sh
curl -X GET http://aloyse:thebest@localhost:5984/
```

Si tout fonctionne, CouchDB renverra une réponse JSON contenant des informations sur le serveur.

Création d'une base de données

```sh
curl -X PUT http://aloyse:thebest@localhost:5984/films
```

Si la base de données est bien créée, CouchDB renvoie une confirmation.

Récupération des informations d'une base de données

```sh
curl -X GET http://aloyse:thebest@localhost:5984/films
```

CouchDB renvoie un JSON contenant les détails de la base de données.

Insertion d'un document

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

Si le document est inséré, CouchDB renvoie un message de confirmation.

> **NB** : Si vous essayez d'insérer un document avec le même identifiant (`doc1`), CouchDB renverra une erreur de **conflit**.

Insertion d'un document via un fichier JSON

Si vous avez un fichier `film.json` contenant :

```json
{
  "titre": "Interstellar",
  "année": 2014,
  "genre": "Science-fiction"
}
```

Vous pouvez l'envoyer avec :

```sh
curl -X POST http://aloyse:thebest@localhost:5984/films -d @film.json -H "Content-Type: application/json"
```

Si aucun identifiant n'est précisé, CouchDB en générera un automatiquement.

Insertion d'une collection de documents

Si vous souhaitez insérer plusieurs films en une seule requête :

```sh
curl -X POST http://aloyse:thebest@localhost:5984/films/_bulk_docs -d @films.json -H "Content-Type: application/json"
```

Le fichier `films.json` doit être structuré ainsi :

```json
{
  "docs": [
    { "titre": "The Dark Knight", "année": 2008, "genre": "Action" },
    { "titre": "Avatar", "année": 2009, "genre": "Science-fiction" }
  ]
}
```

Conclusion

CouchDB est un système de gestion de bases de données NoSQL qui utilise un format JSON flexible et une API REST pour manipuler les données. Il permet de gérer des documents sans schéma rigide, contrairement aux bases

relationnelles. Cependant, cette flexibilité peut entraîner une incohérence si elle est mal gérée.

Dans la prochaine vidéo, nous verrons comment utiliser **MapReduce** avec CouchDB.

Tutoriel2 MapReduce avec CouchDB

2. Installation et Configuration de CouchDB

Vous pouvez vous référer à la première partie du tutoriel pour l'installation de CouchDB.

Connexion à CouchDB

- Utilisation de l'interface graphique de CouchDB.
- Connexion avec les identifiants :
  - **Nom d'utilisateur** : aloyse
  - **Mot de passe** : thebest

Importation de la Collection de Films

La collection de films a été importée en utilisant `curl` . Voici la commande utilisée :

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

3. Exemple 1: Calcul du Nombre de Films par Année

Fonction Map

La fonction Map prend en paramètre un document et émet une clé (année de sortie) et une valeur (titre du film) pour chaque document. Exemple de

transformation :

```javascript
function(doc) {
  emit(doc.year, 1);
}
```

Fonction Reduce

La fonction Reduce permet de regrouper les résultats par année et de calculer le nombre de films par année. La somme des valeurs est utilisée pour obtenir le nombre total de films par année.

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

Résultats Intermédiaires

Les résultats intermédiaires obtenus après l'exécution de la requête MapReduce sont :

```json
{
  "rows": [
    {"key": 1977, "value": 1},
    {"key": 1979, "value": 1},
    {"key": 1992, "value": 1},
    {"key": 2003, "value": 2},
    {"key": 2004, "value": 1},
    {"key": 2005, "value": 3}
  ]
}
```

4. Exemple 2: Calcul du Nombre de Films par Acteur

Fonction Map pour les Acteurs

La fonction Map parcourt le tableau des acteurs pour chaque film et émet une

clé correspondant au prénom et au nom de l'acteur, ainsi qu'une valeur pour chaque film auquel l'acteur a participé.

```javascript
function(doc) {
  if (doc.actors) {
    doc.actors.forEach(function(actor) {
      emit(actor.first_name + " " + actor.last_name, 1);
    });
  }
}
```

Fonction Reduce

La fonction Reduce calcule le nombre de films pour chaque acteur.

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

Résultats

Exemples de résultats obtenus :

```json
{
  "rows": [
    {"key": "Harrison Ford", "value": 3},
    {"key": "Mark Hamill", "value": 1},
    {"key": "Morgan Freeman", "value": 2},
    {"key": "Viggo Mortensen", "value": 2},
    {"key": "Tom Cruise", "value": 2}
  ]
}
```

5. Conclusion

Ce rapport a montré comment utiliser CouchDB pour effectuer des calculs parallèles avec MapReduce. L'exemple a démontré la puissance de CouchDB dans le traitement de grandes quantités de données de manière distribuée. Les

résultats obtenus illustrent comment structurer les données pour obtenir des statistiques pertinentes de manière efficace.