TP4 : Découverte de CouchDB et MapReduce

Guide pratique des bases de données documentaires

Introduction à CouchDB

CouchDB est une base de données NoSQL orientée documents, utilisant le format JSON et accessible via une API REST. Idéale pour : - Prototypage rapide - Applications Web et mobiles - Stockage flexible sans schéma prédéfini

CouchDB expose des fonctionnalités sous forme d'API REST : - **GET** : Récupérer la représentation d'une ressource. - **PUT** : Créer ou mettre à jour une ressource. - **POST** : Envoyer des données à une ressource (ajout ou exécution d'un service). - **DELETE** : Supprimer une ressource.

Caractéristiques clés : - □ Stockage JSON natif - □ Réplication multi-cluster - □ Requêtes via MapReduce - □ Sécurité avec authentification

---

Installation en 2 méthodes

Méthode 1 : Docker (Recommandée)

```
docker run -d --name mycouch \
  -e COUCHDB_USER=admin \
  -e COUCHDB_PASSWORD=secret \
  -p 5984:5984 \
  couchdb:latest
```

Méthode 2 : Installation native

Téléchargez depuis couchdb.apache.org

Vérification :

```
curl http://admin:secret@localhost:5984
# Résultat attendu : {"couchdb":"Welcome","version":"3.3.2"...}
```

Manipulation de base avec cURL

Création de base de données

```
curl -X PUT http://admin:secret@localhost:5984/films
```

Insertion de données

Document unique :

```
curl -X POST http://admin:secret@localhost:5984/films \
  -H "Content-Type: application/json" \
  -d '{"title":"Inception","year":2010}'
```

Insertion en masse :

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

Récupération de document

```
curl -X GET http://admin:secret@localhost:5984/films/doc_id
```

MapReduce par l'exemple

Cas 1 : Statistiques de films par année

Fonction Map :

```
function(doc) {
  if (doc.year && doc.title) {
    emit(doc.year, 1);
  }
}
```

Fonction Reduce :

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

Année	Nombre de films
2020	45
2021	32

Cas 2 : Films par acteur

Fonction Map :

```
function(doc) {
```

```
  doc.actors?.forEach(actor => {
    emit(actor.name, doc.title);
  });
}
```

Fonction Reduce :

```
function(keys, values) {
  return {count: values.length, films: values};
}
```

Exercice : Calcul matriciel avec 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 p
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 Goog
   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 p
   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 $\phi = \sum_{j=1}^{N} M_{ij}$
   suppose que le vecteur W tient en mémoire RAM et est accessible comme variable statique par toutes les fonctions de Map ou de Reduce. Spécifie
   MapReduce qui implante ce calcul.

ennonce

Problème 1 - Norme de vecteurs

Objectif : Calculer ||V|| = √(Σv_i²) pour chaque ligne

Solution Map :

```
function(doc) {
  let sumSquares = Object.values(doc)
    .filter(Number.isFinite)
    .reduce((acc, val) => acc + val**2, 0);

  emit(doc._id, sumSquares);
}
```

Solution Reduce :

```
function(keys, values) {
  return Math.sqrt(values[0]); // Clé unique par document
}
```

Problème 2 - Produit matrice-vecteur

Objectif : Calculer φ_i = Σ(M_ij * w_j)

Données d'exemple :

```
{
  "row": 1,
  "values": [0.2, 0.5, 0.3],
  "vector": [0.4, 0.1, 0.7]
}
```

Solution Map :

```
function(doc) {
  const dotProduct = doc.values
    .reduce((acc, val, idx) => acc + val * doc.vector[idx], 0);

  emit(doc.row, dotProduct);
}
```

Bonnes pratiques

1. Utilisez des ID explicites (`doc_id` significatifs)
2. Indexez les vues fréquemment utilisées
3. Optimisez les fonctions Reduce
4. Utilisez Fauxton pour le débogage : http://localhost:5984/_utils

---

Conclusion

CouchDB combine flexibilité et puissance grâce à : - Stockage JSON natif - Requêtes distribuées via MapReduce - Interface REST simple - Réplication intégrée

Pour aller plus loin : - Documentation officielle - CouchDB en 10 minutes

Amine BOUJEMAAOUI