

TP3 – Exploration de CouchDB

Base de Données NoSQL orientée documents



NICOLAS Ethan

G5SI2
2025–2026

Table des matières

1	Introduction	2
2	Aux origines du MapReduce : matrice de liens et calculs distribués	2
2.1	Énoncé	2
2.2	Réponse 1 : modèle de documents pour représenter M (collection C)	2
2.3	Réponse 2 : MapReduce pour calculer la norme des lignes $\ M_i\ $	3
2.4	Réponse 3 : MapReduce pour calculer $\varphi = M \cdot W$	4
3	Installation et déploiement	5
3.1	Création du volume Docker	5
3.2	Lancement de CouchDB via Docker	5
3.3	Interface Web	5
3.4	Test de connexion HTTP	5
4	Création et gestion des bases	6
4.1	Création d'une base	6
4.2	Insertion de documents	6
4.3	Lecture	6
4.4	Mise à jour	6
5	Définition de vues MapReduce	6
5.1	Comptage des films par année	6
5.2	Index par acteurs	7

1 Introduction

Apache CouchDB est une base de données NoSQL orientée documents reposant sur trois principes fondamentaux : stockage JSON, API REST, et mécanisme MapReduce pour les analyses. Sa réplication incrémentale et sa tolérance aux pannes en font une technologie adaptée aux systèmes distribués et aux environnements multi-nœuds.

Ce TP documente l'installation, la configuration, l'import et la manipulation de données, ainsi que l'utilisation de vues MapReduce.

2 Aux origines du MapReduce : matrice de liens et calculs distribués

2.1 Énoncé

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

1. Proposer un modèle, sous forme de documents structurés, pour représenter une telle matrice (s'inspirer du cas PageRank vu en cours). Soit C la collection ainsi obtenue.
2. La ligne i peut être vue comme un vecteur à N dimensions décrivant la page P_i . Spécifier le traitement MapReduce qui calcule la norme de ces vecteurs à partir des documents de la collection C .

La norme d'un vecteur $V(v_1, v_2, \dots, v_N)$ est :

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

3. Calculer le produit de la matrice M avec un vecteur de dimension N , $W(w_1, w_2, \dots, w_N)$. Le résultat est un vecteur φ défini par :

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

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. Spécifier le traitement MapReduce qui implante ce calcul.

2.2 Réponse 1 : modèle de documents pour représenter M (collection C)

La matrice est énorme et creuse (sparse) : stocker uniquement les liens existants. Modèle "ligne par document" : un document représente une page source P_i (la ligne i) et contient la liste de ses liens sortants non nuls (colonnes j) avec leur poids M_{ij} .

Structure générale d'un document (ligne i).

```
{
  "_id": "page:Pi",
  "type": "page",
  "i": 123,
  "url": "https://example.org/pi",
  "outlinks": [
    { "j": 17, "to": "page:P17", "w": 0.42 },
    { "j": 93, "to": "page:P93", "w": 0.10 }
  ],
  "meta": {
    "crawl_ts": "2025-12-16T00:00:00Z"
  }
}
```

Correspondance avec M .

- La ligne i correspond au document `page:Pi`.
- Chaque élément `outlinks[k]` encode un coefficient non nul : $w = M_{ij}$.
- Si un lien (i, j) n'existe pas, il n'apparaît pas dans `outlinks` (coefficient nul implicite).

2.3 Réponse 2 : MapReduce pour calculer la norme des lignes $\|M_i\|$

La norme d'une ligne i est :

$$\|M_i\| = \sqrt{\sum_{j=1}^N M_{ij}^2}.$$

Avec le modèle sparse, la somme se fait uniquement sur les liens sortants présents.

Principe MapReduce.

- **Map** : pour chaque lien (i, j) de poids $w = M_{ij}$, émettre w^2 sous la clé i .
- **Reduce** : sommer les carrés pour obtenir $S_i = \sum_j M_{ij}^2$.
- **Post-traitement** : calculer $\|M_i\| = \sqrt{S_i}$ après lecture du résultat (ou via un *show/list* côté CouchDB si utilisé).

Fonction Map (CouchDB view).

```
function (doc) {
  if (doc.type !== "page" || !doc.outlinks) return;

  // clé = i, valeur = contribution au carré
  for (var k = 0; k < doc.outlinks.length; k++) {
    var w = doc.outlinks[k].w;
    emit(doc.i, w * w);
  }
}
```

```

// Option: si une page sans outlinks doit apparaître avec norme 0
if (doc.outlinks.length === 0) {
    emit(doc.i, 0);
}
}

```

Reduce. Réduction associative : somme. En CouchDB, utiliser `_sum`.

Reduce = `_sum`

Résultat. La vue retourne, pour chaque i , la valeur :

$$S_i = \sum_j M_{ij}^2 \quad \Rightarrow \quad \|M_i\| = \sqrt{S_i}.$$

2.4 Réponse 3 : MapReduce pour calculer $\varphi = M \cdot W$

On veut :

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

Hypothèse : W tient en RAM et est accessible par les fonctions Map/Reduce comme variable statique (ex. chargée au démarrage de la vue, ou injectée dans l'environnement).

Principe MapReduce.

- **Map** : pour le document (ligne) i , calculer un *partiel* $\sum_{j \in \text{out}(i)} M_{ij} w_j$ en lisant w_j dans $W[j]$. Émettre sous la clé i .
- **Reduce** : somme des partiels (utile si la ligne est fragmentée sur plusieurs docs ; sinon réduction triviale).

Fonction Map (produit matrice-vecteur). Ici, W est supposé disponible via une variable globale `W` indexée par `j`.

```

// Hypothèse: W est accessible globalement, ex.
// var W = { 0: 1.2, 1: -0.3, 17: 0.8, ... };

function (doc) {
    if (doc.type !== "page" || !doc.outlinks) return;

    var acc = 0;
    for (var k = 0; k < doc.outlinks.length; k++) {
        var j = doc.outlinks[k].j;
        var mij = doc.outlinks[k].w;

        var wj = W[j];           // lu en RAM
    }
}

```

```

    if (wj !== undefined) {
        acc += mij * wj;
    }
    // si W[j] absent: contribution 0
}

emit(doc.i, acc);
}

```

Reduce. Somme associative des contributions (CouchDB : `_sum`).

Reduce = `_sum`

Valeur retournée. La vue retourne, pour chaque i :

$$\varphi_i = \sum_{j \in \text{out}(i)} M_{ij} w_j$$

(les coefficients nuls n'étant pas stockés).

3 Installation et déploiement

3.1 Création du volume Docker

```
docker volume create couchdb_data
```

3.2 Lancement de CouchDB via Docker

Les paramètres d'identification administrateur sont fournis par variables d'environnement.

```

docker run \
  --name couchdb \
  -e COUCHDB_USER=ethan \
  -e COUCHDB_PASSWORD=nicolas \
  -p 5984:5984 \
  -v couchdb_data:/opt/couchdb/data \
  -d couchdb

```

3.3 Interface Web

Accessible via :

http://localhost:5984/_utils/

3.4 Test de connexion HTTP

```
curl -X GET http://ethan:nicolas@localhost:5984
```

```
{"couchdb":"Welcome","version":"3.5.1"... "vendor":{"name":"The Apache Software Foundation"}}
```

4 Création et gestion des bases

4.1 Création d'une base

```
curl -X PUT http://ethan:nicolas@localhost:5984/films
```

4.2 Insertion de documents

Insertion unitaire

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

Insertion par lot

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

4.3 Lecture

```
curl -X GET http://ethan:nicolas@localhost:5984/films/<id_doc>
```

4.4 Mise à jour

```
curl -X PUT http://ethan:nicolas@localhost:5984/films/<id_doc> \  
-H "Content-Type: application/json" \  
-d '{"_rev":"1-xxx","titre":"Dune","annee":2021}'
```

5 Définition de vues MapReduce

5.1 Comptage des films par année

Fonction Map

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

Fonction Reduce

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

5.2 Index par acteurs

Map

```
function (doc) {  
  if (doc.actors) {  
    doc.actors.forEach(a => emit(a, doc.title));  
  }  
}
```

Reduce

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