Katell GOUZERH ING3A INFO
2025/2026

TP4 : MapReduce avec CouchDB

MapReduce est un paradigme de calcul distribué conçu pour faciliter le traitement de grandes quantités de données. Il repose sur une séparation claire des responsabilités entre plusieurs étapes successives.

La première étape, appelée Map, consiste à parcourir les documents d'entrée et à produire des couples *(clé, valeur)* à partir des informations pertinentes. La seconde étape, Reduce, regroupe l'ensemble des valeurs associées à une même clé afin de produire un résultat agrégé. Une étape optionnelle, nommée Finalize, permet d'effectuer un calcul final, par exemple une moyenne ou un ratio.

Ce modèle est largement utilisé dans les bases de données NoSQL orientées documents. Dans ce TP, il est étudié à travers deux systèmes : MongoDB et CouchDB, qui proposent des implémentations différentes du concept.

Partie 1 : MapReduce avec MongoDB

MongoDB est une base de données NoSQL orientée documents. Les données sont stockées sous forme de documents JSON (ou BSON) regroupés en collections. Chaque document peut contenir des champs simples, des tableaux ou des structures imbriquées.

MongoDB propose deux mécanismes principaux pour l'agrégation des données : le pipeline d'agrégation et MapReduce. Dans ce TP, l'utilisation de MapReduce permet d'illustrer le fonctionnement détaillé de ce paradigme.

Dans MongoDB, MapReduce est exécuté de manière ponctuelle. Le traitement est déclenché manuellement et le résultat est stocké dans une collection de sortie. Les fonctions Map, Reduce et Finalize sont écrites en JavaScript.

```
db.films.mapReduce(mapFunction, reduceFunction, {
out: "collection_resultat",
finalize: finalizeFunction
})
```

1. Nombre total de films

```
var map = function() { emit("total", 1); };
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex1_total_films" });
```

2. Nombre de films par genre

```
var map = function() {
if (this.genre)
this.genre.forEach(g => emit(g, 1));
};
```

```
var reduce = function(key, values) {
return Array.sum(values);
};

db.films.mapReduce(map, reduce, { out: "ex2_films_par_genre" });
```

3. Nombre de films par réalisateur

```
var map = function() { emit(this.director, 1); };
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex3_films_par_realisateur" });
```

4. Nombre de films par année

```
var map = function() { emit(this.year, 1); };
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex4_films_par_annee" });
```

5. Nombre moyen de films par réalisateur

```
var map = function() { emit(this.director, 1); };
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex5_films_par_realisateur" });
```

La moyenne est ensuite calculée à partir de la collection résultat.

6. Genres les plus représentés

```
var map = function() {
if (this.genre)
this.genre.forEach(g => emit(g, 1));
};
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex6_genres_dominants" });
```

7. Acteurs les plus présents

```
var map = function() {
if (this.actors)
this.actors.forEach(a => emit(a, 1));
};
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex7_acteurs_frequents" });
```

8. Nombre de films par acteur

```
var map = function() {
if (this.actors)
this.actors.forEach(a => emit(a, 1));
};
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex8_films_par_acteur" });
```

9. Films par couple acteur/genre

```
var map = function() {
if (this.actors && this.genre)
this.actors.forEach(a => {
this.genre.forEach(g => emit({ actor: a, genre: g }, 1));
});
};
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex9_acteur_genre" });
```

10. Films par décennie

```
var map = function() {
var decade = Math.floor(this.year / 10) * 10;
emit(decade, 1);
};
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex10_films_par_decennie" });
```

11. Durée moyenne des films

```
var map = function() {
emit("avg", { sum: this.runtime, count: 1 });
};

var reduce = function(key, values) {
var total = { sum: 0, count: 0 };
values.forEach(v => {
total.sum += v.sum;
total.count += v.count;
});
return total;
};

var finalize = function(key, value) {
return value.sum / value.count;
```

```
};

db.films.mapReduce(map, reduce, { out: "ex11_duree_moyenne", finalize: finalize });
```

12. Films par pays

```
var map = function() { emit(this.country, 1); };
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex12_films_par_pays" });
```

13. Films par langue

```
var map = function() { emit(this.language, 1); };
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex13_films_par_langue" });
```

14. Films par classification

```
var map = function() { emit(this.rated, 1); };
var reduce = function(key, values) { return Array.sum(values); };

db.films.mapReduce(map, reduce, { out: "ex14_films_par_classification" });
```

15. Analyse personnalisée

```
var map = function() {
emit(this.year, { total: this.runtime, count: 1 });
};


var reduce = function(key, values) {
var res = { total: 0, count: 0 };
values.forEach(v => {
res.total += v.total;
res.count += v.count;
});
return res;
};


var finalize = function(key, value) {
return value.total / value.count;
};


db.films.mapReduce(map, reduce, { out: "ex15_analyse_libre", finalize: finalize });
```

Partie 2 : CouchDB et MapReduce

CouchDB est une base de données NoSQL orientée documents, conçue pour fonctionner nativement avec le web. Les données sont stockées au format JSON et accessibles uniquement via une API REST basée sur HTTP. Chaque document possède un identifiant unique ainsi qu'un champ de version (`_rev`) permettant la gestion de la concurrence.

Contrairement à MongoDB, CouchDB ne propose pas l'exécution ponctuelle de requêtes MapReduce. Le paradigme MapReduce est utilisé exclusivement à travers des vues persistantes, stockées dans des design documents.

MongoDB utilise MapReduce comme un outil analytique lancé à la demande, dont les résultats sont stockés dans des collections. CouchDB adopte une approche différente : les fonctions MapReduce servent à construire des index consultables en permanence. Ainsi :

- MongoDB → analyse ponctuelle
- CouchDB → indexation persistante

On commence par deployer CouchDB avec Docker :

```
docker volume create couchdb_data

docker run \
        --name couchdb \
        -e COUCHDB_USER=quingjie \
        -e COUCHDB_PASSWORD=ktl \
        -p 5984:5984 \
        -v couchdb_data:/opt/couchdb/data \
        -d couchdb
```

Ensuite on créée et on gère la base de donnée :

```
curl -X PUT http://quingjie@localhost:5984/films
curl http://quingjie@localhost:5984/_all_dbs
```

On insere des documents :

```
curl -X POST http://quingjie@localhost:5984/films \ //document unique
        -H "Content-Type: application/json" \
        -d '{"title":"Inception","year":2010,"genre":["Sci-Fi"],"actors":["Leonardo
DiCaprio"],"runtime":148}'

curl -X POST http://quingjie@localhost:5984/films/_bulk_docs \ //insertion en masse
        -H "Content-Type: application/json" \
        -d @films_couchdb.json
```

Les vues MapReduce sont définies dans des design documents. Un design document contient un ensemble de vues, chacune étant composée d'une fonction Map et éventuellement d'une fonction Reduce.
Créons maintenant un design document :

```
curl -X PUT http://quingjie@localhost:5984/films/_design/films_views \
-H "Content-Type: application/json" \
-d '{
        "views": {
                "films_par_annee": {
                        "map": "function(doc) { if(doc.year) emit(doc.year, 1); }",
                        "reduce": "_count"
                }
        }
}
```

Vue 1 : Nombre de films par année

Fonction Map

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

Fonction Reduce

```
_count
```

Requête de consultation

```
curl http://quingjie@localhost:5984/films/_design/films_views/_view/films_par_annee?
group=true
```

Vue 2 : Indexation des films par acteur

Fonction Map

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

Requête

```
curl http://quingjie@localhost:5984/films/_design/films_views/_view/films_par_acteur
```

Vue 3 : Nombre de films par genre

Fonction Map

```
function(doc) {
        if (doc.genre)
        doc.genre.forEach(g => emit(g, 1));
}
```

Fonction Reduce

```
_sum
```

Requête

```
curl http://quingjie@localhost:5984/films/_design/films_views/_view/films_par_genre?group=true
```