3ème année cycle d'ingénieur – Spécialité Informatique

TP4 – MapReduce avec MongoDB

Réalisé par :
ER-RACHDI Ilias

Encadrés par :
M. YOUCEF

Année Scolaire : 2025/2026

1 Introduction

Ce TP a pour objectif d'introduire le paradigme **MapReduce** à travers MongoDB, en s'appuyant sur la collection de films étudiée lors des TP précédents. MapReduce permet d'effectuer des traitements distribués sur de grands volumes de données en séparant le calcul en deux phases principales : une phase de *map*, qui produit des paires clé-valeur intermédiaires, et une phase de *reduce*, qui agrège ces résultats.

L'objectif est de comprendre la logique de ce modèle de calcul, son intérêt pour l'analyse de données volumineuses, ainsi que sa mise en œuvre concrète dans MongoDB.

2 Principe général de MapReduce

Dans MongoDB, un traitement MapReduce repose sur trois fonctions :

- **Map** : parcourt chaque document de la collection et émet des paires clé-valeur,
- **Reduce** : regroupe les valeurs associées à une même clé afin de produire un résultat agrégé,
- **Finalize** (optionnelle) : permet d'appliquer un traitement final sur le résultat du reduce.

La commande générale utilisée dans ce TP est la suivante :

```
db.films.mapReduce(mapFunction, reduceFunction, {
out: "resultat"
})
```

3 Exercices MapReduce

Dans cette section, les réponses sont présentées sous la forme de traitements MapReduce complets, avec les fonctions JavaScript extttmap, extttreduce et, lorsque nécessaire, extttfinalize, conformément au format attendu.

Question 1 : Compter le nombre total de films

Map

```
function() {
emit("total_films", 1);
}
```

Reduce

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

Question 2 : Compter le nombre de films par genre

Map

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

Reduce

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

Question 3 : Compter le nombre de films par réalisateur

Map

```
function() {
if (this.director) {
emit(this.director, 1);
}
}
```

Reduce

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

Question 4 : Compter les acteurs uniques

Map

```
function() {
if (this.actors) {
this.actors.forEach(a => emit(a, 1));
}
}
```

Reduce

```
function(key, values) {
return 1;
}
```

Question 5 : Nombre de films par année de sortie

Map

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

Reduce

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

Question 6 : Note moyenne par film

Map

```
function() {
if (this.grades) {
const mean = this.grades.reduce((a, b) => a + b.score, 0) / this.grades.
    length;
emit(this.title, mean);
```

```
}
}
```

Reduce

```
function(key, values) {
return Array.sum(values) / values.length;
}
```

Question 7 : Note moyenne par genre

Map

```
function() {
if (this.grades && this.genre) {
const avg = this.grades.reduce((a, b) => a + b.score, 0) / this.grades.length
    ;
this.genre.forEach(g => emit(g, avg));
}
}
```

Reduce

```
function(key, values) {
return Array.sum(values) / values.length;
}
```

Question 8 : Note moyenne par réalisateur

Map

```
function() {
if (this.director && this.grades) {
const avg = this.grades.reduce((a, b) => a + b.score, 0) / this.grades.length
    ;
emit(this.director, avg);
}
}
```

Reduce

```
function(key, values) {
return Array.sum(values) / values.length;
}
```

Question 9 : Film avec la note maximale

Map

```
function() {
if (this.grades) {
const max = Math.max(...this.grades.map(g => g.score));
emit(this.title, max);
}
}
```

Reduce

```
function(key, values) {
return Math.max(...values);
}
```

Question 10 : Nombre de notes supérieures à 70

Map

```
function() {
if (this.grades) {
const count = this.grades.filter(g => g.score > 70).length;
emit("notes_sup_70", count);
}
}
```

Reduce

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

Question 11 : Acteurs par genre sans doublons

Map

```
function() {
if (this.genre && this.actors) {
this.genre.forEach(g => {
this.actors.forEach(a => emit(g, a));
});
}
}
```

Reduce

```
function(key, values) {
return Array.from(new Set(values));
}
```

Question 12 : Acteurs apparaissant dans le plus grand nombre de films

Map

```
function() {
if (this.actors) {
this.actors.forEach(a => emit(a, 1));
}
}
```

Reduce

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

Question 13 : Classement des films par grade majoritaire

Map

```
function() {
if (this.grades) {
const freq = {};
this.grades.forEach(g => {
freq[g.grade] = (freq[g.grade] || 0) + 1;
});
const major = Object.keys(freq).reduce((a, b) => freq[a] > freq[b] ? a : b);
emit(major, this.title);
}
}
```

Reduce

```
function(key, values) {
return values;
}
```

Question 14 : Note moyenne par année

Map

```
function() {
if (this.grades && this.year) {
const avg = this.grades.reduce((a, b) => a + b.score, 0) / this.grades.length
    ;
emit(this.year, avg);
}
}
```

Reduce

```
function(key, values) {
return Array.sum(values) / values.length;
}
```

Question 15 : Réalisateurs avec une moyenne supérieure à 80

Map

```
function() {
if (this.director && this.grades) {
const avg = this.grades.reduce((a, b) => a + b.score, 0) / this.grades.length
    ;
emit(this.director, { sum: avg, count: 1 });
}
}
```

Reduce

```
function(key, values) {
let s = 0, c = 0;
values.forEach(v => { s += v.sum; c += v.count; });
return { sum: s, count: c };
}
```

Finalize

```
function(key, value) {
const avg = value.sum / value.count;
return avg > 80 ? avg : null;
}
```

4 Conclusion

Ce TP a permis de mettre en pratique le modèle MapReduce dans MongoDB à travers des exemples concrets d'analyse de données. Les différents exercices montrent comment ce paradigme permet de traiter efficacement des collections volumineuses en répartissant le calcul.

Bien que MapReduce soit aujourd'hui souvent remplacé par des pipelines d'agrégation plus performants, il reste un outil pédagogique pertinent pour comprendre les principes fondamentaux du calcul distribué.