Chapitre 4

Map/Reduce

Dans ce chapitre, nous allons étudier le comportement d'une requête Map/Reduce sous MongoDB. Pour ce faire, nous allons commencer par s'intéresser au Map, puis à la partie Reduce.

4.1 Reduce

Dans cette section, nous allons modifier la fonction `reduce`.

4.1.1 Pour chaque source de commentaire, donner le nombre moyen de mots ;
Attention, la moyenne n'est pas une fonction associative (contrairement à la somme) - problème de reduce local et global.
Correction : Naturellement, nous aurions tendance à écrire ce type de programme Map/Reduce (moyenne des valeurs dans le reduce).

```
mapFunction = function () {
        for(i=0; i<this.reviews.length;i++)
                        emit(this.reviews[i].source, this.reviews[i].wordsCount);
};
reduceFunction = function (key, values) {
        return Array.avg(values);
};
queryParam = {"query":{}, "out":{"inline":true}};
db.paris.mapReduce(mapFunction, reduceFunction, queryParam);
```

Toutefois, pour des raisons d'optimisation (minimisation des communications réseaux) des reduce locaux (bloc de données) sont calculés avant les reduce globaux (ensemble des blocs de données). Ce problème survient dès que la fonction d'agrégat n'est pas associative :

— Associative : min, max, sum

— Non-Associative : avg, count, append

Pour remédier à ce problème, il faut implémenter les fonctions `Map` et `Reduce` avec le même schéma de sortie (ici : {titles : [x, y, z]}). Le `Reduce` s'occupe de décomposer la liste de liste.

```
mapFunction = function () {
        for(i=0; i<this.reviews.length;i++){
                review = this.reviews[i];
                emit(review.source, { "avg" : review.wordsCount,
                        "sum" : review.wordsCount, "nb": 1  });
        }
};
reduceFunction = function (key, values) {
        sum = 0; nb = 0;
        for (i=0; i< values.length ; i++){
                sum += values[i].sum;
                nb += values[i].nb;
        }
        return {"avg" : sum / nb, "sum" : sum, "nb" : nb};
};
queryParam = {"query":{}, "out":{"inline":true}};
db.paris.mapReduce(mapFunction, reduceFunction, queryParam);
```

Nous constaterons de nettes différences de valeurs dans les moyennes calculées dans les deux versions.

La bonne pratique pour écrire un programme Map/Reduce est de faire en sorte que la sortie du Map ait le même schéma que la sortie du reduce. Il faut donc pour ce faire, donner le résultat local, et les valeurs permettant de calculer le résultat global (ou partiellement). Quelque soit le nombre d'étapes de reduce appliqué.

4.1.2 Même question, mais en produisant la note moyenne (rating), min et max ;
Correction : Pour être sûr de bien traiter le reduce de reduce, il faut préparer le format de sortie dès la fonction 'map'

```
mapFunction = function () {
        for(var i=0; i<this.reviews.length;i++){
                review = this.reviews[i];
                rating = parseInt(review.rating);
                if(rating)
                        emit(review.source, { "avg" : rating, "sum" : rating, "nb": 1,
                                "min" : rating, "max" : rating  });
        }
};
reduceFunction = function (key, values) {
        sum = 0; nb = 0; min = values[0].min; max = values[0].max;
        for (i=0; i< values.length ; i++){
                sum += values[i].sum;
                nb += values[i].nb;
                if(values[i].min < min) min = values[i].min;
                if(values[i].max > max) min = values[i].max;
        }
        return {"avg":sum/nb, "sum":sum, "nb":nb, "min":min, "max":max};
};
```

4.1.3 Donner la liste distincte des services proposés ;
Correction :

```
mapFunction = function () {
        if(this.services && this.services.length > 0)
                        emit(null, {"services" : this.services} );
        }
}
reduceFunction = function (key, values) {
        output = new Array ();
        for(i=0; i<values.length;i++){
                for(j=0;j<values[i].services.length;j++){
                        service = values[i].services[j];
                        if (!Array.contains (output, service))
                                output.push(service);
                }
        }
        return {"services" : output};
};
```

Attention, le reduce et le re-reduce vont créer des listes de listes de services. Il faut donc recomposer la liste de services distincts (contains et push).

4.1.4 Pour chaque source, nombre de langues de commentaires différents par type de source ;
Correction : Il faut, pour chaque source, donner la liste des langues distinctes, et la taille de cette liste.

```
mapFunction = function () {
        sources = {};
        for(var i=0;i<this.reviews.length;i++){
                review = this.reviews[i];
                if(sources[review.source]){
                        src = sources[review.source];
                        if(!Array.contains(src.langs, review.language)){
                                src.langs.push(review.language);
                                src.nb++;
                        }
                } else
                        src = {"langs" : [review.language], "nb" : 1};
                sources[review.source] = src;
        }

        for(key in sources)
                emit (key, sources[key]);
};
var reduceFunction = function (key, values) {
        var distinct = values[0].nb;
        var langs = values[0].langs;
        for(i=1 ; i<values.length ; i++){
                l = values[i];
                for(j=0; j< l.langs.length; j++)
                if(!Array.contains(langs, l.langs[j])){
                        langs.push(l.langs[j]);
                        distinct++;
                }
        }
        return {"langs" : langs, "nb" : distinct};
};
```

4.1.5 Pour chaque source de commentaire, donner la moyenne du nombre de commentaires par année. Réutiliser la requête "source&annee", puis calculer la moyenne avec "aggregate" ;
Correction :

```
mapFunction = function () {
        for(var i=0;i<this.reviews.length;i++){
                review = this.reviews[i];
                if(review.time) annee = review.time.substring(0,4);
                else                annee = null;
                emit({"src":review.source, "y" : annee}, 1);
        }
};
reduceFunction = function (key, values) {
        return Array.sum(values);};
queryParam = {"query":{}, "out":"result"};
db.paris.mapReduce(mapFunction, reduceFunction, queryParam);

db.result.aggregate([{$group : {"_id" : "$_id.src", "moy" : {$avg : "$value" }}}]);
```

Il faut d'abord calculer par couple "source&année", puis faire la moyenne par source.

4.1.6 Le refaire, mais en une seule requête "map/reduce" ;
Correction :

```
mapFunction = function () {
        for(var i=0;i<this.reviews.length;i++){
                review = this.reviews[i];
                if(review.time) annee = review.time.substring(0,4);
                else annee = null;
                annees = {};
                annees[annee] = 1;
                annees["avg"] = 1;
                emit(review.source, annees);
        }
};
reduceFunction = function (key, values) {
        annees = {};
        for(i=0 ; i< values.length ; i++){
                v = values[i];
                for(key in v){
                        if(key != "avg"){
                                if(annees[key])
                                        annees[key] += v[key];
                                else
                                        annees[key] = v[key];
                        }
                }
        }

        nb_values = 0;
        sum_values = 0;
        for(key in annees){
                sum_values += annees[key];
                nb_values++;
        }
        if(nb_values > 0)
                annees["avg"] = sum_values / nb_values;

        return annees;
};
queryParam = {"query":{}, "out":{"inline" : 1}};
db.paris.mapReduce(mapFunction, reduceFunction, queryParam);
```

Il est possible également de ne faire qu'un seul emit par langue dans le map :

```
mapFunction = function () {
        langs = {};

        for(var i=0;i<this.reviews.length;i++){
                review = this.reviews[i];
                if(review.time) annee = review.time.substring(0,4);
                else annee = null;
                if(langs[review.source]){
                        annees = langs[review.source];
                        if(annees[annee])
                                annees[annee] ++;
                        else
                                annees[annee] = 1;
                        annees["avg"]++;
                } else {
                        annees = {};
                        annees[annee] = 1;
                        annees["avg"] = 1;
                }
                langs[review.source] = annees;
        }

        for(key in langs)
                emit (key, langs[key]);
};
```

Nous pourrons remarquer une diminution du nombre de ”emit” (86903 → 17303)