Models and Systems for Big Data

Map Reduce & MongoDB

Nacéra Seghouani

Computer Science Department, CentraleSupélec
Laboratoire de Recherche en Informatique, LRI LaHDAK group
nacera.seghouani@centralesupelec.fr

2019-2020

Introduction

- MongoDB provides a MapReduce framework, easy to experiment locally
- The functions Map and Reduce are implemented in javascript
- Drivers `http://docs.mongodb.org/ecosystem/drivers/`

  `Python, Ruby, Java, Javascript (Node.js), C++, C#, PHP, Perl, Scala...`
- Syntax

  `http://docs.mongodb.org/ecosystem/drivers/syntax-table/`

MapReduce - Javascript

```
var mapFunction = function () {
   if( this.age > 30 && this.job.contains("MI6") )
   emit (this.address.city, this.age);
}
```

emit: (key, value), this: current document

```
var reduceFunction = function (key, values) {
  return Array.length(values);
}
```

returns a unique value aggregated from the list of "values" (for a given key)

```
var queryParam = {query : {}, out : "result_set"}
db.users.mapReduce (mapFunction, reduceFunction, queryParam);
db.result_set.find();
```

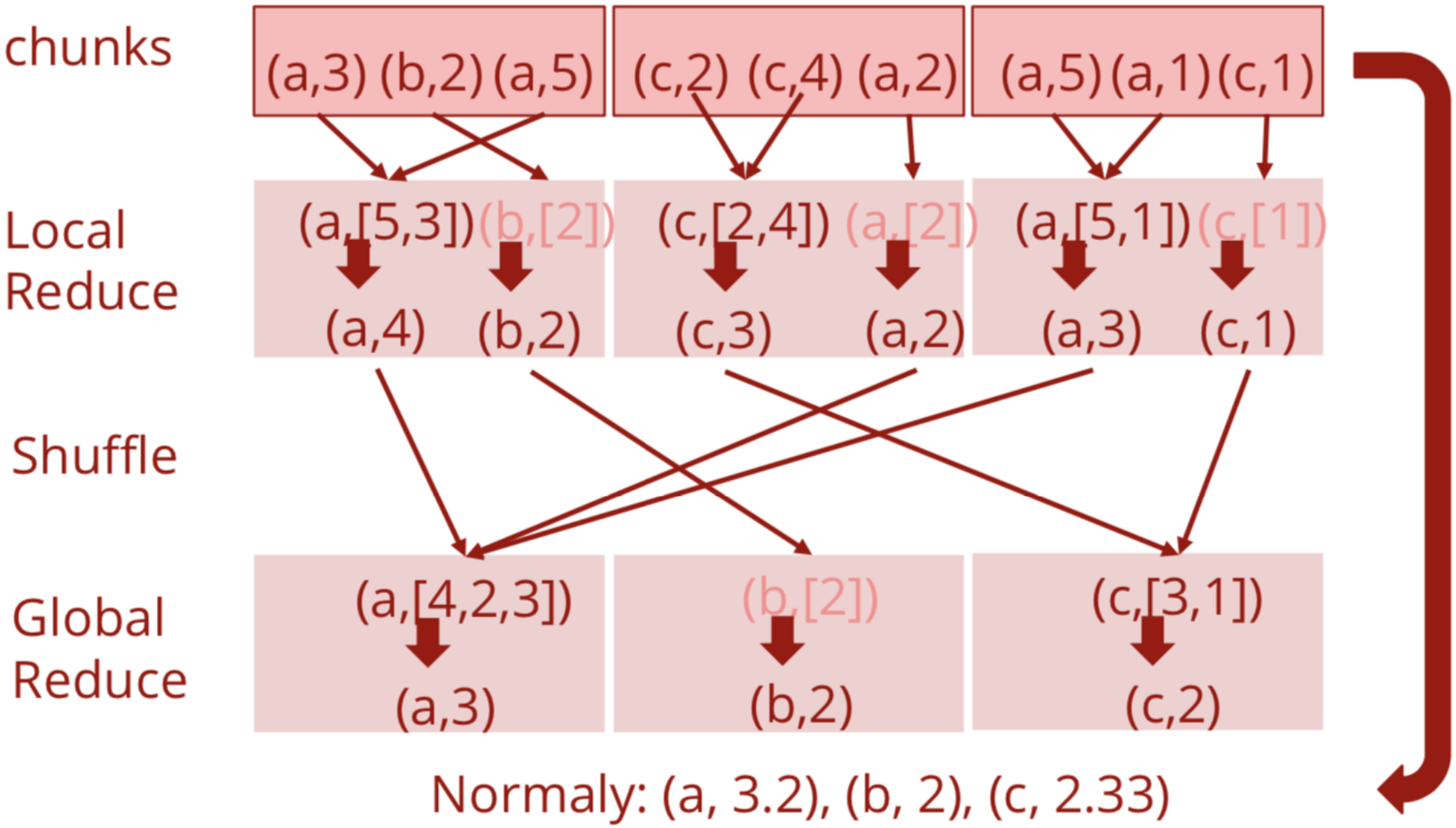
query: filter before applying the map
out: output collection

CentraleSupélec

MapReduce - Javascript

- Map
  - Several emit per map (key/value)
  - Applied on every document
- Reduce
  - Optimization : local and global
  - No reduce if only one value
- Shuffle
  - Group values by keys (from emit)

MapReduce - Javascript



MapReduce - Example 1

Work on `moviesEmbedded` collection

```
var mapDirector = function() {
     emit(this.director._id, this.title);
};
var reduceDirector = function(directorId, titles) {
      var res = new Object();
      res.director = directorId;
      res.movies = titles;
      return res;
};
var queryParam = {query : {}, out : "result_set"}
db.moviesEmbedded.mapReduce(mapDirector, reduceDirector, {out: {"inline": 1}})
db.moviesEmbedded.mapReduce(mapDirector, reduceDirector, queryParam)
db.moviesEmbedded.mapReduce(mapDirector, reduceDirector, {out: {"inline": 1}},
{query: {"country": "USA"}})
```

MapReduce - Example 2

Work on `moviesReferences` and `Artists` collections, but the two collections are merged in `moviesReferencesArtists` collection because of `mapReduce` limitation.

```
var mapJoin = function() {
 // does the document id key contain "artist"
 if (this._id.indexOf("artist") != -1) {
 // if yes, add the type
 this.type="artist";
 // produce key id artist,
 emit(this._id, this);
 }
 else {
 // if no, it's a movie, add the type
 this.type="movie";
 // simplify the document remove summary and actors
 delete this.summary;
 delete this.actors;
 // produce a pair with key
 emit(this.director._id, this); }
};
```

MapReduce - Joining two collections

Work on `moviesReferences` and `Artists` collections, but the two collections are merged in `moviesReferencesArtists` collection because of `mapReduce` limitation.

```
var reduceJoin = function(id, items) {
  var director = null, movies={result: []}
  // look for artist in this list
 for (var idx = 0; idx < items.length; idx++) {
   if (items[idx].type=="artist") {
   director = items[idx];
    }
 }
// now, 'director' contains the artist: assign to movies
 for (var idx = 0; idx < items.length; idx++) {
  if (items[idx].type=="movie" && director != null) {
         items[idx].director = director;
         movies.result.push (items[idx]);
  }
 }
 return movies;
};
db.moviesReferencesArtists.mapReduce(mapJoin, reduceJoin, {out: {"inline": 1}});
```