Pivot Using MapReduce

Rui SONG

rui.song@polytechnique.edu

This document is a report for the homework of course "Big Data Framework" for Master 2 Data Science of University of Paris Saclay. The content explains how to use mapReduce to find the transpose of a geant matrix.

Suppose that we have a file that contains the content of a matrix (each element of the matrix is separated by a comma in the file), we will split this file into different pieces and pass each to a map machine.

Before splitting, we insert the row number in the beginning of that row (followed by a comma). Suppose that this file has M rows, and we have n1 map machines. Then each file passed to the map machine will contain at least [M/n1] rows of the matrix.

For simplicity, suppose the matrix has 3 columns, and the origin matrix is like below:

$$\begin{bmatrix} titi & toto & tutu \\ bibi & bobo & bubu \end{bmatrix}$$

The values pass to the map machines is like below:

1, titi, toto, tutu
2, bibi, bobo, bubu

The pseudocode of the map function can be written like below:

```
map (String key, String value):
    // key: document name
    // value: document content
    for each line l in value:
        // l: first value is the row number in the matrix,
        //    the rest of the values are the elements of in that row of the matrix
        //    different values are separated by a comma
```

```
        String[] elements = l.split(“,”);
        for (int i =1; i < elements.length; i++):
        EmitIntermediate(i, { elements[0], elements[i]})
```

The results of the map phase will be like below:

```
{1: {1: titi}}, {2: {1: toto}}, {3: {1: tutu}}
{1: {2: bibi}}, {2: {2: bobo}}, {3: {2: bubu}}
```

Then for values that have the same key (the same column number) we sort them according to its row number:

```
{1: [{1: titi}, {2: bibi}]}
{2: [{1: toto}, {2: bobo}]}
{3: [{1: tutu}, {2: bubu}]}
```

After sorting, we will emit all the values that has the same key (row number in the matrix) to the reduce function. The pseudocode of the map function can be written like below:

```
reduce (String key, Iterator values):
    // key: column number
    // values: a list of “row number”-“elements” pairs
    list ls;
    for v in values:
        ls.append(v.value);
    Emit(key, ls);
```

The results after the reduce phase is shown as below:

```
{1: [titi, bibi]}
{2: [toto, bobo]}
{3: [tutu, bubu]}
```