MapReduce

1 Computing averages

We are given a dataset that contains the average monthly temperature measurements over the course of some years. More precisely, the dataset is stored in a CSV file, where each row corresponds to a monthly measurement and the columns contain the following values: year, month, average temperature in the month.

```
1980,1,5
1980,2,2
1980,3,10
1980,4,14
1980,5,17
....
1981,1,2
1981,2,1
1981,3,3
1981,4,10
....
```

We intend to get the average monthly temperature for each year.

> **Exercise**
> **Exercise 1.1.** Write a MapReduce algorithm that generates key-value pairs $(year, average\_temperature)$.

Solution

> map: $(year, month, temperature) \rightarrow (year, temperature)$
> reduce: $(year, temps) \rightarrow (year, sum(temps)/len(temps))$
> - $temps$ is the list of all temperatures in the same $year$.
> - $sum(temps)$ sums all the elements in the list $temps$.
> - $len(temps)$ gives the length of the list $temps$.

Suppose now that we have a large CSV file stored in a distributed file system (e.g., HDFS), containing a series of measurements in the format "Year, Month, Day, Minute, Second, Temperature". We can have up to one measurement per second in some years. Like before, we'd like to compute key-value pairs (year, average_temperature) by using a MapReduce algorithm.

> **Exercise**
> **Exercise 1.2.** What is the maximum number of measurements in a year?

Solution

Since we can have up to one measurement per second, the maximum number of measurements M_{max} for a certain year is given by the following formula:

$$M_{max} = 365 \times 24 \times 60 \times 60 \approx 31.5 \times 10^6$$

Exercise
Exercise 1.3. Considering the answer to the previous question, discuss the efficiency of the first implementation of the algorithm.

Solution

Since there might be up to 31 million values associated with a key, the bottleneck of the computation would be the shuffle operation, since we need to copy a high number of (key,value) pairs from the mappers to the reducers.
Also, a reducer might have to loop over a huge list of values in order to compute their average.

Exercise
Exercise 1.4. Based on the answer to the previous question, propose a better implementation to handle the CSV file.

Solution

map: $(year, mo, d, mi, sec, temperature) \rightarrow (year, temperature)$
combine: $(year, temps) \rightarrow (year, (sum(temps), len(temps)))$
reduce: $(year, [(s_i, l_i),\ i = 1 \dots n]) \rightarrow (year, \frac{\sum_{i=1}^{n} s_i}{\sum_{i=1}^{n} l_i})$

- $temps$ is the list of all temperatures in the same $year$.
- $sum(temps)$ sums all the elements in the list $temps$.
- $len(temps)$ gives the length of the list $temps$.

2 Computing average and standard deviation

We consider again the large CSV file with a series of measurements in the format "Year, Month, Day, Minute, Second, Temperature". We now intend to generate a series of key-value pairs (year, (avg_temperature, std_deviation)).

We can express the standard deviation of n values x_i $(1 \leq i \leq n)$ with two different equations.

The first equation is as follows:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}}$$

The second equation is as follows:

$$\sigma = \sqrt{\overline{x^2} - \overline{x}^2} = \sqrt{\frac{\sum_{i=1}^{n}(x_i)^2}{n} - \left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^2}$$

Exercise
Exercise 2.1. Which equation of the standard deviation is more appropriate in a Map-Reduce algorithm? Why?

Solution

The second equation is more appropriate because it allows the computation of the sum of the elements and of the square of the elements step by step by using map and combine together. Instead, if we use the first equation, we need first to compute the average and then use it to compute the variance.

Exercise
Exercise 2.2. Propose a MapReduce implementation to compute the average and the standard deviation of the temperatures for each year.

Solution

map: $(year, mo, d, mi, sec, temperature) \rightarrow (year, temperature)$
combine: $(year, T) \rightarrow (year, (sum(T), sum(T^2), len(T)))$
reduce: $(year, [(s_i, sq_i, l_i),\ i = 1 \dots n]) \rightarrow (year, (\mu, \sigma))$
where:

- T is the list of all temperatures in the same $year$.
- $sum(T)$ sums all the elements in the list T.
- $T^2 = [x^2 | x \in T]$
- $len(T)$ gives the length of the list T.
- $\mu = \sum_{i=1}^{n} s_i / \sum_{i=1}^{n} l_i$
- $\sigma = \sqrt{(\sum_{i=1}^{n} sq_i / \sum_{i=1}^{n} l_i) - \mu^2}$

3 Common friends in a social network

Consider a social network described by a graph encoded in a text file. Each line of the file is a list of identifiers separated by commas. For instance, the line A, B, C, D means that A is friend with B, C and D. An excerpt of the file looks like as follows:

```
A,B,C,D
B,A,D
C,A,D
D,A,B,C
...
```

We suppose that the friendship relation is symmetric: (A, B) implies (B, A).

We want to obtain the list of the common friends for each pair of individuals:

```
(A, B), [D]
(A, C), [D]
(A, D), [B, C]
(B, C), [D]
(B, D), [A]
(C, D), [A]
```

As an additional constraint, we want to represent a couple only once and avoid to represent the symmetric couple. In other words, if we output (A, B), we don't want to output (B, A).

Exercise
Exercise 3.1. Propose a MapReduce implementation to find the common friends in a social network satisifying the given constraints.

Solution

map: $(x, F) \rightarrow [((u, v), x)\ \forall (u, v) \in F \mid u < v]$
reduce: $[(u, v), LCF] \rightarrow [(u, v), LCF]$
where:
- x is the first item in a line.
- F is the list containing the items in a line except the first one (x's friends).
- LCF is the list of all individuals that are friends with both u and v.

We note that the reduce function is the identity.

4 Creating an inverted index

We have a collection of n documents in a directory and we want to create an **inverted index**, one that associates each word to the list of the files the word occurs in. More precisely, for each word, the inverted index will have a list of the names of the documents that contain the word.

Exercise
Exercise 4.1. Propose a MapReduce implementation to create an inverted index over a collection of documents.

Solution

The input to the map will be a key-value pair, where the key is the name of a file f and the value is the content C of the file.
map: $(f, C) \rightarrow [(w, f)\ \forall w \in C]$
reduce: $(w, L) \rightarrow (w, L)$
where L is the list of the files containing the word w.
We note that the reduce function is the identity.
Note also that in the map function we can add instructions to preprocess the text. For example, we can eliminate some words that are not useful in the index (e.g., the stopwords) or remove special symbols.