Project 3: Big Data Analysis in Hadoop System

Yunfei Liu
School of Computer Science, Shanghai Jiao Tong University, Shanghai, Minhang, 800 Dongchuan RD.
liuyunfei@sjtu.edu.cn

***Abstract*—This is the third project report of final assignments of Course Parallel Computing and Parallel Programming Autumn, 2020.**

I. Task

In this project, we need to implement weatherdata program using MapReduce. We must use weatherdata program to get the statistics of the highest and lowest temperature for all the files in the data directory, respectively.

II. Data Preprocessed

We can use R language to get weatherdata directly with following commands.

```
> install.packages("devtools")
> library("devtools")
> install_github("Ram-N/weatherData")
> library(weatherData)
```

Then we can use write function to save the data into txt files.

```
> write.table(⟨DATASET⟩, "⟨DATASET⟩.txt",
sep = ' ', row.names = FALSE, col.names
= FALSE)
```

Each term of data record consists of a timestamp and the corresponding temperature value logged in Fahrenheit. Related scale information of gathered temperature dataset is as listed in Table I.

TABLE I
The statistical information of dataset.

Document	Scale
London2013.txt	20639
Mumbai2013.txt	18328
NewYork2013.txt	10199
SFO2012.txt	9670
SFO2013.txt	9507

III. Hadoop

We use HUAWEI cloud to build the hadoop cluster. There are 1 namenode hadoop-master and 2 datanodes: hadoop-worker1, hadoop-worker2. This process is very complex. I mainly refer to this [1]article. Since it's not the key part of this project, I will not give the details here.

[1]https://blog.csdn.net/sunxiaoju/article/details/85222290

After we successfully build hadoop cluster. We can use "jps" command to check whether it works.

Now we put data into hadoop file system. First we make directories "input" and "output", where we put input files and output files.

```
> hdfs dfs -mkdir /input
> hdfs dfs -mkdir /output
```

Then we put weather data into /input folder

```
> hdfs dfs -put weatherdata/* /input/
```

IV. MapReduce

MapReduce can be summarized into five steps as shown in Figure. 1 during the running process:

- Input Step. The input data is obtained from hdfs and partitioned as the input of map.
- Map Step. Process resolves a record of a certain input format into one or more records.
- Shuffle Step. Control of intermediate data as the input of reduce.
- Reduce Step. Merge the data of the same key.
- Output Step. Output to the specified directory according to the format.

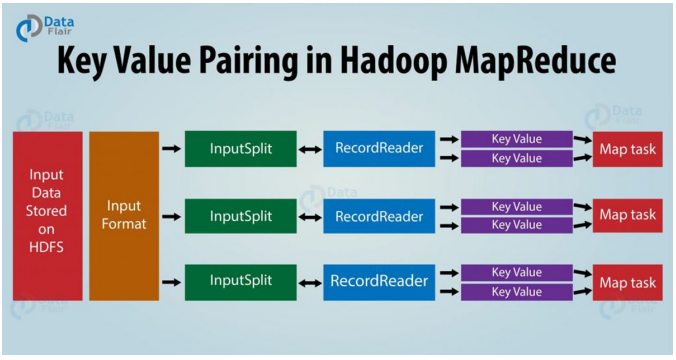

Fig. 1. Steps of MapReduce in Hadoop.

A. Input

When the program loads temperature data file, it will split data file into small pieces and each piece is allocated to a map task. This process is implemented by MapReduce framework. We can use **FileInputFormat** and **TextInputFormat** in **org.apache.hadoop.mapreduce.lib.input**

```
FileInputFormat.setInputPaths(job, new
Path(args[0]));
job.setInputFormatClass(TextInputFormat
.class);
```

B. Map

We need to define a map function to process the input data from input step. The input is ¡key, value¿ pairs. Each value is just the temperature in a time step. Because we save the data with " " as the separate symbol, here we can split the input value with " ". Then we change String data into Double type and store the minimum/maximum value.

C. Shuffle

The output ⟨`key, value`⟩ pair of map step then go into shuffle step. The shuffle step optimizes the intermediate data and distributes the data to each reduce process. In this step, it will participate, sort, and combine the input data.

D. Reduce

The input of reduce step is a iterators of key value pairs. So we just read the value one by one and compare with the max value to find the global max value.

E. Output

We save results on the disk. The step can be finished by the built in output-format `FileOutputFormat` and `TextOutputFormat` defined in `org.apache.hadoop.mapreduce.lib.output` which is similar to input step.

V. Experiments

The experiments are conducted on the hadoop cluster we built on HUAWEI Cloud. Overall results are recorded and listed in Table. II, with statistical running time as shown in Table. III.

TABLE II
THE HIGHEST AND LOWEST TEMPERATURE STATISTICS.

Data	Max Temperature	Min Temperature
London2013	91.4 °F	23.0 °F
Mumbai2013	102.2 °F	53.0 °F
NewYork2013	99.0 °F	12.0 °F
SFO2012	91.9 °F	36.0 °F
SFO2013	88.0 °F	35.1 °F

TABLE III
AVERAGE TIME SPENT BY MAP AND REDUCE TASKS.

Data	Total Time Spend (AVG)	
	all map tasks	all reduce tasks
London2013	3388 ms	3521 ms
Mumbai2013	3390 ms	3566 ms
NewYork2013	3428 ms	3397 ms
SFO2012	3314 ms	3587 ms
SFO2013	3454 ms	3390 ms

VI. Demo Link

The demo for this project is upload to jbox, you can follow the link below and check the running process.

- The demo link: https://jbox.sjtu.edu.cn/l/b1kNaQ.