ENGR 516 — Assignment 1

We have known below tasks in the class,

1. Set up a cluster with Hadoop Distributed File System and run examples.
2. On top of HDFS, set up the cluster with MapReduce programming framework.
3. Run examples of MapReduce programs.

Based on our examples, we will develop our own MapReduce program to analysis a simple log file. The following figure shows the structure of the log file.

```
221.13.63.157 - - [28/Jan/2019:17:09:59 +0800] GET /activity.php HTTP/1.1 "200" 88 "-" "GPS\xE7\x8A\xB6\xE6\x80\x81 1.5.31 rv:1535 (iPhone; iOS 11.4.1; zh_CN)" "-"
2001:470:c:829::2 - - [28/Jan/2019:17:10:02 +0800] GET //weather/weather_data_open.php?city= HTTP/1.0 "200" 0 "-" "-" "-"
31.167.82.193 - - [28/Jan/2019:17:10:02 +0800] GET /weather/weather_data_new.php?city= HTTP/1.1 "200" 31 "-" "GPS Status 1.5.31 rv:1535 (iPhone; iOS 12.1.2; en_SA@calendar=gregorian)" '
221.13.63.157 - - [28/Jan/2019:17:10:09 +0800] GET //save_token.php?
token=c1934c06%207589cf9b%205fae6ae0%20e5baf5e5%2007bc1f66%20e72e10e6%20b663a0e7%20cf670cbd&os=IOS&version=11.4.1&type=iPhone8,1&city=%E9%81%B5%E4%B9%89&loc=%3C+27.65227959,+106.8889991
%2065.00m%20(speed%20-1.00%20mps%20/%20course%20-1.00)%200%202019/1/28%20%E4%B8%AD%E5%9B%BD%E6%A0%87%E5%87%86%E6%97%B6%E9%97%B4%20%E4%B8%8B%E5%8D%885:10:08&device_id=5356CE59-7643-45E3-
A81EA417408E&app_version=1.2 HTTP/1.1 "200" 33 "-" "GPS\xE7\x8A\xB6\xE6\x80\x81 1.5.31 rv:1535 (iPhone; iOS 11.4.1; zh_CN)" "-"
221.13.63.157 - - [28/Jan/2019:17:10:09 +0800] GET /weather/weather_data_new.php?city=%E9%81%B5%E4%B9%89 HTTP/1.1 "200" 260 "-" "GPS\xE7\x8A\xB6\xE6\x80\x81 1.5.31 rv:1535 (iPhone; iOS
zh_CN)" "-"
116.239.210.101 - - [28/Jan/2019:17:10:30 +0800] GET /activity.php HTTP/1.1 "200" 88 "-" "GPS Status 1.5.31 rv:1535 (iPad; iOS 10.2; en_CN)" "-"
117.170.237.223 - - [28/Jan/2019:17:10:41 +0800] GET /activity.php HTTP/1.1 "200" 75 "-" "GPS\xE7\x8A\xB6\xE6\x80\x81 1.5.31 rv:1535 (iPhone; iOS 12.1.1; zh-Hans_HK)" "-"
201.227.226.136 - - [28/Jan/2019:17:10:43 +0800] GET /activity.php HTTP/1.1 "200" 88 "-" "GPS Status 1.5.31 rv:1535 (iPhone; iOS 12.1.2; en_PA)" "-"
2001:470:c:829::2 - - [28/Jan/2019:17:10:45 +0800] GET //weather/weather_data_open.php?city=Panam%C3%A1 HTTP/1.0 "200" 0 "-" "-" "-"
```

Each line is a record of visit, which consists of **IP Address**, **Time**, **Type of HTTP Request**, **Requested File**, **HTTP Version** and **Status**, etc.

Example Programs

We have provided two examples that related to this lab.

- logstat: It counts the number of visits for each IP address in the log file.
- logstat2: It counts the number of visits for each IP address in the same hour.

As discussed in the lectures, MapReduce programming framework seperates the data and operation (two stages). It uses Hadoop Stream, which represents by **sys.stdin** in Python and **Writable, Text** in Java.

In Map phase, we have to process the raw files and extract the related information, line, IP.

In the Reduce phase, we start counting the records based on the same IP addresses. After that, we can sort the result and print it out. As Fig. 5 and 6 present, the Map Phase for logstat2 is different than the previous version since we need to consider the **time**. Since we have processed data at Map Phase, the intermediate data of Map is already at the granularity of a hour. Therefore, the Reduce Phase is the same as logstat.

Lab 1 Assignment: Part 1 and 2

The given programs of logstat and logstat2 were written in both JAVA and Python.

Lab 1 consists of the following two parts.

```python
import re
import sys

pat = re.compile('(?P<ip>\d+.\d+.\d+.\d+).*?"\w+ (?P<subdir>.*?) ')
for line in sys.stdin:
    match = pat.search(line)
    if match:
        print '%s\t%s' % (match.group('ip'), 1)
```

Figure 1: Map Phase of logstat in Python

```java
public class LogStatMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {

    protected void map(LongWritable key, Text value,Context context)
            throws IOException, InterruptedException {
        Pattern p = Pattern.compile("(\\d+\\.\\d+\\.\\d+\\.\\d+).*?");
        Matcher matcher = p.matcher(value.toString());
        while(matcher.find()){
            context.write(new Text(matcher.group(1)), new IntWritable(1));
        }
```

Figure 2: Map Phase of logstat in Java

```python
dict_ip_count = {}

for line in sys.stdin:
    line = line.strip()
    ip, num = line.split('\t')
    try:
        num = int(num)
        dict_ip_count[ip] = dict_ip_count.get(ip, 0) + num

    except ValueError:
        pass

sorted_dict_ip_count = sorted(dict_ip_count.items(), key=itemgetter(0))
for ip, count in sorted_dict_ip_count:
    print '%s\t%s' % (ip, count)
```

Figure 3: Reduce Phase of logstat in Python

```
public class LogStatReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {

    protected void reduce(Text arg0, Iterable<IntWritable> arg1,Context
    arg2)
            throws IOException, InterruptedException {
        int sum = 0;
        for(IntWritable i : arg1){
            sum += i.get();
        }
        arg2.write(arg0, new IntWritable(sum));
```

Figure 4: Reduce Phase of logstat in Java

```
import re
import sys

pat =
re.compile('(?P<ip>\d+.\d+.\d+.\d+).*?\d{4}:(?P<hour>\d{2}):\d{2}.*? ')
for line in sys.stdin:
    match = pat.search(line)
    if match:
        print '%s\t%s' % ('[' + match.group('hour') + ':00' + ']' +
        match.group('ip'), 1)
```

Figure 5: Map Phase of logstat2 in Python

```
public class LogStatMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {

    protected void map(LongWritable key, Text value,Context context)
            throws IOException, InterruptedException {
        Pattern p = Pattern.compile("(\\d+\\.\\d+\\.\\d+\\.\\d+).*?");
        Matcher matcher = p.matcher(value.toString());
        while(matcher.find()){
            context.write(new Text(matcher.group(1)), new IntWritable(1));
        }
```

Figure 6: Map Phase of logstat2 in Java

1. Output the top-3 IP addresses with the granularity of an hour
    - You should read the two examples.
    - Develop your code based on examples. The program may take more than one round of MapReduce.
2. Make your program like a *database* search
    - Your program should be able to accept parameters from users, such as 0-1, which means from time 00:00 to 01:00, and output the top-3 IP addresses in the given time period.
    - Run it along with three other examples, WordCount, Sort, Grep, at the same time, and test fair and capacity schedulers.

Grading Rubric

(80%) Part 1;
(15%) Part 2;
(5%) Report about the your design and experiments, please include screenshots for running your code on the cloud;

Submission

You upload your submission in Canvas by the end of February 24.