

빅데이터 최신기술

서울 공기질 데이터 분석

목차

1. 과제 수행 방법

- 1.1. Google Cloud Platform
- 1.2. Problem1
- 1.3. Problem2
- 1.4. Problem3
- 1.5. Problem4

2. 실행 화면

- 2.1. Google Cloud Platform

3. 결과

- 3.1. Problem1
- 3.2. Problem2
- 3.3. Problem3
- 3.4. Problem4

소프트웨어학부

20163162

차윤성

1. 과제 수행 방법

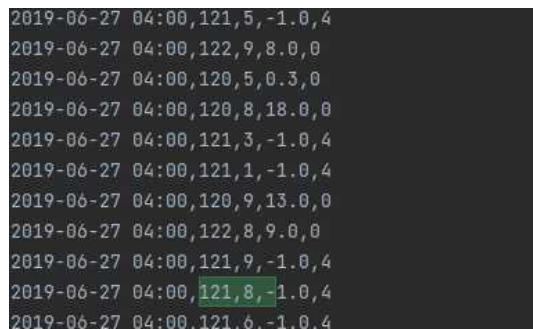
1.1 Google Cloud Platform

- HDFS(Hadoop Distributed File System)을 사용하기 위한 플랫폼
- Master Node : Worker Node = 1 : 3
- SCP 명령을 통해 Master Node로 Local File 전송
\$ > scp -P 2222 <File_Path> <User_Name>@<Master_Node_IP_Addr>
- SSH 접속 가능
\$ > ssh <Master_Node_IP_Addr> -p 2222 -l <User_Name>
- SSH 접속 후 hdfs dfs -put 명령을 통해 hdfs에 data 파일 저장
\$ > hdfs dfs -put <Master_Node's_File_Path>
- hadoop jar 명령을 실행하면 hdfs에 저장되어있는 data 파일로 Map Reduce 과정 실행
\$ > haddop jar <Jar_File> <Driver_Class_Path>
-Dmapreduce.job.reduces=<Num_Reducer> <HDFS_Data_Path>

1.2 Problem 1

- Map

- Input <Key, Value> Type: <Object, Text>
: Key - 읽어 들인 파일, Value - 파일에서 읽어온 한 줄
- Output <Key, Value> Type: <Text, DoubleWritable>
: Key - 지역 코드 + 공기질 종류 코드, Value - 공기질 코드의 값



```
2019-06-27 04:00,121,5,-1.0,4
2019-06-27 04:00,122,9,8.0,0
2019-06-27 04:00,120,5,0.3,0
2019-06-27 04:00,120,8,18.0,0
2019-06-27 04:00,121,3,-1.0,4
2019-06-27 04:00,121,1,-1.0,4
2019-06-27 04:00,120,9,13.0,0
2019-06-27 04:00,122,8,9.0,0
2019-06-27 04:00,121,9,-1.0,4
2019-06-27 04:00,121,8,-1.0,4
2019-06-27 04:00,121,6,-1.0,4
```

<그림1 - data.csv>

- 이 중 statusCode 값이 0이 아니면 normal data가 아니기 때문에 0이 아닌 경우는 Reduce로 넘겨주지 않도록 처리함
- 이 때, 읽어 들인 공기질 종류의 코드가 8이 아니면, 어떠한 처리도 하지 않음
(itemCode == 8 -> PM10)

- Reduce

- Input <Key, Value> Type: <Text, Iterable<DoubleWritable>>
: Key - 지역코드 + 공기질 종류 코드
: Value - 지역코드에서 관측된 모든 PM10 수치값들
- Output <Key, Value> Type: <Text, Text>
: Key - 지역코드 + 공기질 종류 코드
: Value - PM10 수치의 평균값 + 최댓값 + 최솟값
- 지역코드 별로 관측된 PM10 수치의 평균값과 최댓값과 최솟값을 계산

1.2 Problem2

- Map

- Input <Key, Value> Type: <Object, Text>
 - : Key - 읽어 들인 파일, Value - 파일에서 읽어온 한 줄
- Output <Key, Value> Type: <Text, DoubleWritable>
 - : Key - 지역 코드 + 공기질 종류 코드, Value - 공기질 코드 값
- 이 중 statusCode값이 0이 아니면 Reduce로 넘기지 않도록 처리함
(statusCode == 0 -> normal data, normal data만 처리하도록 함)
- 또한 공기질 종류의 코드가 8이나 9가 아니면, Reduce로 넘기지 않도록 처리

- Reduce

- Input <Key, Value> Type: <Text, DoubleWritable>
 - : Key - 지역 코드 + 공기질 종류 코드, Value - 공기질 코드 값
- Output <Key, Value> Type: <Text, IntWritable>
 - : Key - 지역코드 + 공기질 종류(String)
 - : Value - 공기질 코드 기준 별 공기질 “좋음”이 관측된 횟수
- Input Key값에서 itemCode를 가져와 itemCode별로 처리
 - itemCode == 8(PM10) : 0 < itemValue <= 30 인 경우 카운트를 올림
 - itemCode == 9(PM2.5) : 0 < itemValue <= 15 인 경우 카운트를 올림
- Output file's Format
 - : <StationCode> <ItemName> <Count>
 - StationCode: 지역 코드
 - ItemName: PM10 또는 PM2.5
 - Count: 각 공기질 코드 별 기준으로 공기질 “좋음”이 관측된 횟수
- 이후 local에서의 작업을 통해 PM10 기준으로 공기질 “좋음”이 가장 많이 관측된 횟수와 지역 코드, PM2.5 기준으로 공기질 “좋음”이 가장 많이 관측된 횟수와 지역 코드를 확인하는 프로그램 작성

1.3 Problem3

- Map

- Input <Key, Value> Type: <Object, Text>
 - : Key - 읽어 들인 파일, Value - 파일에서 읽어온 한 줄
- Output <Key, Value> Type: <Text, Text>
 - : Key - 시간 + 지역 코드, Value - 공기질 종류 코드 + 공기질 코드 값
- 이 중 statusCode값이 0이 아니면 Reduce로 넘기지 않도록 처리함
(statusCode == 0 -> normal data, normal data만 처리하도록 함)

- Reduce

- Input <Key, Value> Type: <Text, Text>
 - : Key - 시간 + 지역 코드, Value - 공기질 종류 코드 + 공기질 코드 값
- Output <Key, Value> Type: <Text, Text>
 - : Key - 시간 + 지역 코드, Value - 모든 <공기질 종류 + 값>의 쌍
- 단순히 공기질 코드 case별로 value를 받아와 하나의 string으로 만들어 저장

Active Jobs

Job ID	Name	State	Map Progress	Maps Total	Maps Completed	Reduce Progress	Reduces Total	Reduces Completed
job_1622535582895_0005	Seoul-0.1.jar	RUNNING	<div></div>	1	1	<div></div>	3	0

Showing 1 to 1 of 1 entries

<그림 4 - Working Problem 2 in GCP with 3 reducers>

- [그림4]를 확인하면, 현재 Map Progress는 완료되고 Reduce Progress가 진행중이며, Reduce Task의 개수가 3개임을 확인할 수 있다.
- Hadoop 실행 명령

\$>hadoop jar Seoul-0.1.jar seoul.p#.Problem# -Dmapreduce.job.reduces=3 \data.csv

- #: 1 ~ 4 중 하나의 값, p#과 Problem #에 붙는 숫자는 같아야 함

Job Overview

Job Name:

Seoul-0.1.jar

User Name:

vaite714

Queue:

default

State:

SUCCEEDED

Uberized:

false

Submitted:

Tue Jun 01 14:20:12 UTC 2021

Started:

Tue Jun 01 14:20:20 UTC 2021

Finished:

Tue Jun 01 14:20:37 UTC 2021

Elapsed:

17sec

Diagnostics:

Average Map Time

7sec

Average Shuffle Time

3sec

Average Merge Time

0sec

Average Reduce Time

0sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Tue Jun 01 14:20:13 UTC 2021	kmu-cluster-5e6d-w-0.asia-northeast1-c.c.hallowed-oven-313406.internal:8042	/gateway/default/jobhistory/logs

Task Type	Total	Complete
Map	1	1
Reduce	3	3

Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	3

<그림 5 - Worked Problem 2 in GCP with 3 reducers>

- [그림5]를 확인하면, Map&Reduce Progress가 모두 종료되었고, 모든 과정이 성공적으로 Complete 된 것을 확인할 수 있다.

```
vaite714@kmu-cluster-5e6d-m: ~
Combine output records=0
Reduce input groups=24
Reduce shuffle bytes=64407953
Reduce input records=3775778
Reduce output records=24
Spilled Records=7551556
Shuffled Maps =3
Failed Shuffles=0
Merged Map outputs=3
GC time elapsed (ms)=566
CPU time spent (ms)=19900
Physical memory (bytes) snapshot=1795678208
Virtual memory (bytes) snapshot=17400176640
Total committed heap usage (bytes)=1718616064
Peak Map Physical memory (bytes)=871440384
Peak Map Virtual memory (bytes)=4943607296
Peak Reduce Physical memory (bytes)=334864384
Peak Reduce Virtual memory (bytes)=4355182592
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=124382733
File Output Format Counters
Bytes Written=3877
vaite714@kmu-cluster-5e6d-m: $
```

<그림 6 - 모든 문제의 Map&Reduce를 모두 실행 한 후 화면>

```

vaite714@kmu-cluster-5e6d-m: ~
vaite714@kmu-cluster-5e6d-m: $ hdfs dfs -ls
Found 7 items
-rw-r--r-- 2 vaite714 hadoop 124382733 2021-06-01 08:24 data.csv
drwxr-xr-x - vaite714 hadoop 0 2021-06-01 14:18 data.csv_p1.out
drwxr-xr-x - vaite714 hadoop 0 2021-06-01 14:20 data.csv_p2.out
drwxr-xr-x - vaite714 hadoop 0 2021-06-01 14:26 data.csv_p3.out
drwxr-xr-x - vaite714 hadoop 0 2021-06-01 14:29 data.csv_p4.out
-rw-r--r-- 2 vaite714 hadoop 8257 2021-05-31 10:27 sample.txt
drwxr-xr-x - vaite714 hadoop 0 2021-05-31 10:35 sample.txt.out
vaite714@kmu-cluster-5e6d-m: $

```

<그림 7 - 실행 후 HDFS>

- [그림 7]을 확인해보면, HDFS에 p1~p4까지의 결과가 저장되어있는 것을 확인할 수 있다.

3. 결과

3.1 Problem 1

1	101	8	37.733367431246805	289.0	3.0
2	102	8	38.04298686620679	296.0	3.0
3	103	8	35.90305508780371	330.0	3.0
4	104	8	42.10383539752297	423.0	1.0
5	105	8	42.61197038255862	401.0	3.0
6	106	8	43.92973977695167	389.0	3.0
7	107	8	44.3444359109099	411.0	3.0
8	108	8	41.40356203197733	340.0	3.0
9	109	8	39.77476783324547	326.0	3.0
10	110	8	39.2093618057811	414.0	3.0
11	111	8	44.38306386418755	421.0	3.0
12	112	8	39.002576615264495	322.0	2.0

<그림 8.1 - p1의 local 결과 일부>

106	8	43.92973977695167	389.0	3.0
109	8	39.77476783324547	326.0	3.0
112	8	39.002576615264495	322.0	2.0
115	8	42.79501330828245	293.0	3.0
118	8	39.6592073862525	329.0	3.0
121	8	44.95263464580038	385.0	3.0
124	8	42.44131683248403	426.0	1.0
101	8	37.733367431246805	289.0	3.0
104	8	42.10383539752297	423.0	1.0
107	8	44.3444359109099	411.0	3.0
110	8	39.2093618057811	414.0	3.0
113	8	40.84886350113055	354.0	1.0
116	8	43.93804243008679	389.0	3.0

<그림 8.2 - p1의 HDFS 결과 일부>

- HDFS 결과 확인 명령어

\$ > hdfs dfs -cat data.csv_p1.out/*

- local에서 test한 결과와 hadoop을 사용해 HDFS에 저장된 결과의 총합이 같음을 볼 수 있다.

3.2 Problem 2

1	101	PM10	11766
2	101	PM2.5	10110
3	102	PM10	11960
4	102	PM2.5	10076
5	103	PM10	12601
6	103	PM2.5	9897
7	104	PM10	10277
8	104	PM2.5	8643
9	105	PM10	9964
10	105	PM2.5	9644
11	106	PM10	9299
12	106	PM2.5	7672

<그림 9.1 - p2의 local 결과 일부>

101	PM2.5	10110
102	PM10	11960
104	PM2.5	8643
105	PM10	9964
107	PM2.5	9410
108	PM10	9971
110	PM2.5	9854
111	PM10	8665

<그림 9.2 - p2의 HDFS 결과 일부>

- HDFS 결과 확인 명령어

```
$ > hdfs dfs -cat data.csv_p2.out/*
```

- local에서 test한 결과와 hadoop을 사용해 HDFS에 저장된 결과의 총합이 같음을 볼 수 있다.

```
C:\Users\CYS\.jdk\corretto-1.8.0_292\bin\java.exe ...
PM10 기준 공기질 " 좋음"이 12601번으로 가장 많이 관측된 지역 코드는 103입니다.
PM2.5 기준 공기질 " 좋음"이 11228번으로 가장 많이 관측된 지역 코드는 112입니다.
Process finished with exit code 0
```

<그림 10 - P2Output.java 결과>

- [그림 10]은 local 환경에서 따로 생성한 프로그램으로, Problem 2 Driver를 실행한 후 저장된 Data를 이용해 PM10 / PM2.5 별로 공기질 “ 좋음”이 가장 많이 관측된 지역 코드를 보여주는 프로그램이다.
- PM10 기준으로 지역 코드 103이, PM2.5 기준으로 지역 코드 112가 공기질 “ 좋음”이 가장 많이 관측된 지역이다.

3.3 Problem 3

```
2019-12-31 23:00 125 <O3: 0.005,
CO: 0.5,
NO2: 0.037000000000000005,
PM10: 27.0,
PM2.5: 18.0,
SO2: 0.003>
```

<그림 11.1 - p3의 local 결과 일부>

```
2019-12-31 23:00 125 <O3: 0.005,
PM10: 27.0, SO2: 0.003, PM2.5: 18.0, CO:
0.5, NO2: 0.037000000000000005>
```

<그림 11.2 - p3의 HDFS 결과 일부>

- HDFS 결과 확인 명령어

```
$ > hdfs dfs -cat data.csv_p3.out/*
```

- local에서 test한 결과와 hadoop을 사용해 HDFS에 저장된 결과의 총합이 같음을 볼 수 있다.
- 또한 각 공기질 측정 기준 별 평균값들이 잘 연산되었음을 확인할 수 있다.

3.4 Problem 4

```
09:00    <SO2: 0.004522865680674974,  
NO2: 0.03155268817204228,  
CO: 0.5715915551680613,  
O3: 0.019093188608599015,  
PM10: 43.25349857006673,  
PM2.5: 24.246250237326752>
```

<그림 12.1 - p4의 local 결과 일부>

```
09:00    <SO2: 0.004522865680674994,  
NO2: 0.03155268817204239, CO: 0.5715  
915551680609, O3: 0.0190931886085989  
88, PM10: 43.25349857006673, PM2.5:  
24.246250237326752>
```

<그림 12.2 - p4의 HDFS 결과 일부>

- HDFS 결과 확인 명령어

```
$ > hdfs dfs -cat data.csv_p4.out/*
```

- local에서 test한 결과와 hadoop을 사용해 HDFS에 저장된 결과의 총합이 같음을 볼 수 있다.
- 또한 시간대별로 공기질 측정 기준들의 평균값들이 잘 연산되었음을 확인할 수 있다.

=> 결론: local에서 test한 Map Reduce 시스템과 Hadoop과 GCP를 이용한 Map Reduce 시스템 모두 같은 결과를 주지만, HDFS에 저장된 결과의 경우 Reducer의 숫자만큼 나뉘진 결과 파일을 cat 명령을 통해 합쳐서 확인하는 것이기 때문에 local과 달리 data가 잘 정렬되어 있진 않다.