REPORT

국민대학교
KOOKMIN UNIVERSITY

과목명 | 빅데이터최신기술

담당교수 | 박하명 교수님

학과 |

학년 |

학번 | 20181703

이름 | 평선호

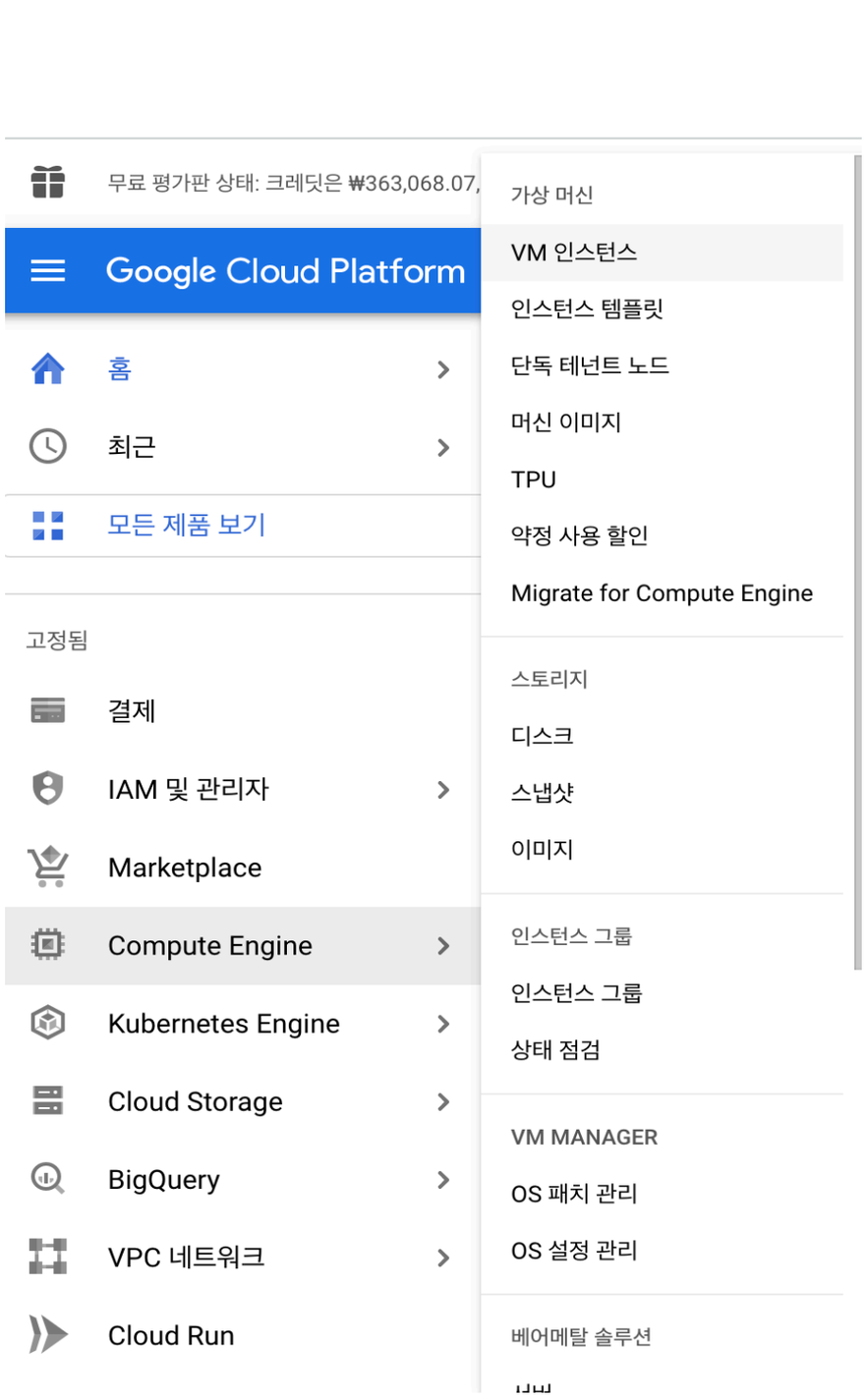
제출일 |

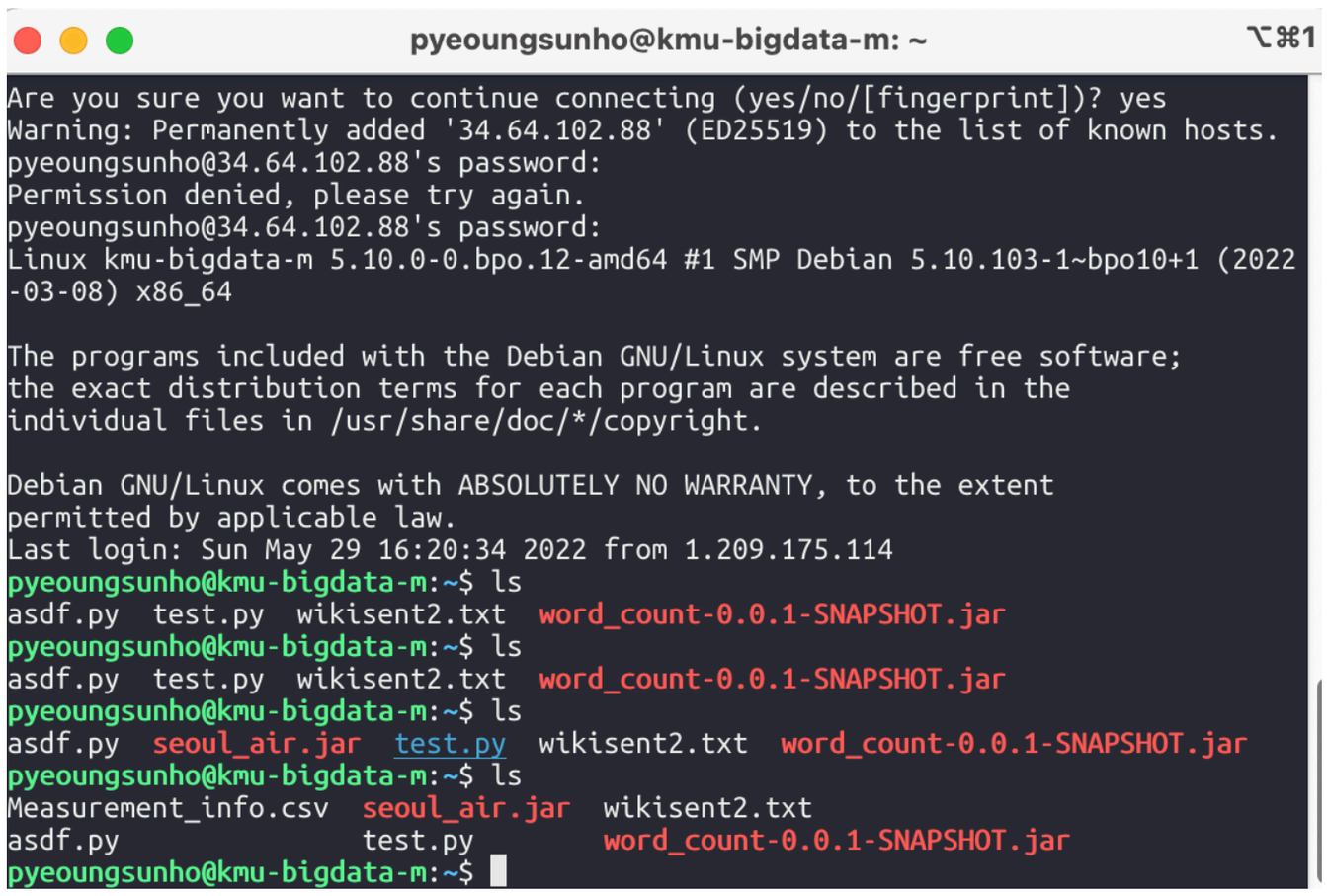
목차

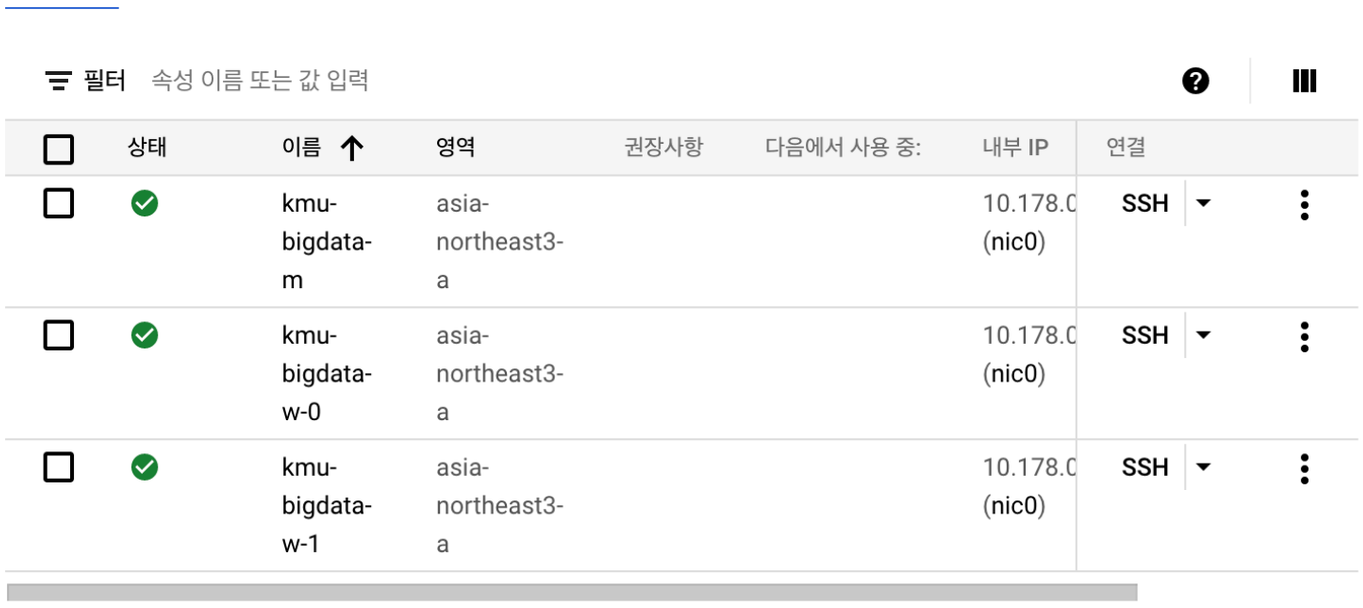
1. 문제1) 각 지역(station code)별로 평균, 최대, 최소 PM10 측정치 구하기
2. 문제2) PM10, PM2.5 기준으로 공기의 질이 ‘좋음’ 수준이 가장 많이 측정된 지역은 어디인지 찾기
3. 데이터 변환하기 -> 각 <시간,지역> 별로 모든 종류의 측정치 모아서 저장하기
4. 시간대를 기준으로 평균 공기질 구하기 (So2,No2.Co,O3,PM10,PM 2.5 한꺼번에 구하기)

<초기 세팅>

Google cloud platform을 이용하여 HDFS(Hadoop Distributed File System)을 사용하였다.
ssh명령어를 사용하여 해당 가상 머신의 인스턴스의 ip주소로 접근 하였고 scp 명령어를 통해 나의 PC에 있는 file들을 Master node로 전송하였다.
이후 접속후 $ >hdfs dfs -put 명령어를 통해 hdfs에 data 파일을 저장하였다.







문제 1 해결 과정

-MAP

- input: < Key, Value> Type: <Object, Text>

key로 해당 파일을 읽어 들인 후 Value는 파일에서 한줄 씩 읽어온다.

이후 StringTokenizer를 통해 값을 한개씩 읽어온다. 그러나 읽어오는 Measurement_info.csv파일에 column명이 있기때문에 예외처리를 해줘야 한다. 이때 startwith 를 사용하여 column명 예외처리를 진행 하였다.
읽어오는 값들 중 StatusCode값이 0 이 아니면 정상적인 데이터가 아니기 때문에 0이 아니면 Reduce로 넘겨주지 않는다. 또한 공기질 종류의 코드가 8이 아니면 처리하지 않는다.

```java
// set Key
while(st.hasMoreTokens()) {
    String date = st.nextToken(); // 2017-01
    if (date.startsWith("Measurement")) return;
    String stationCode = st.nextToken();    // 101
    if (stationCode.startsWith("Station")) return;

    String itemCode = st.nextToken();           // 1
    if( itemCode.startsWith("Item")) return;

    // set Value
    double itemValue = Double.parseDouble(st.nextToken());  // 0.004

    String statusCode = st.nextToken();     // 0
    if (statusCode.startsWith("Instrument")) return;

if(Integer.parseInt(itemCode) == 8) {
    if(Integer.parseInt(statusCode) == 0) {

        ok.set(stationCode + "\t" + itemCode);

        ov.set(itemValue);

        // emit
        context.write(ok, ov);
    }
}
```

P1 Mapper

문제 1 해결 과정

-Reduce

- input: < Key, Value> Type: <Text, Iterable<DoubleWritable>>

  reduce에서 묶은 값들을 연산과정을 통해 처리한다.

각 지역 코드 별로 관측된 PM수치의 평균값과 최댓값과 최솟값을 계산한다.

```java
ublic class P1Reducer extends Reducer<Text, DoubleWritable, Text, Text> {

    Text ov = new Text();

    @Override
    protected void reduce(Text key, Iterable<DoubleWritable> values,Reducer<

        double min = Double.POSITIVE_INFINITY;
        double max = Double.NEGATIVE_INFINITY;

        // set Value(average)
        double sum = 0;
        int cnt = 0;
        for(DoubleWritable d: values){

            double v = d.get();

            if(v < min) min = v;
            if(max < v) max = v;

            sum += v;
            cnt += 1;
        }

        double avg = sum / cnt;

        // set Output Format
        ov.set(avg + "\t" + max + "\t" + min);

        // emit
        context.write(key, ov);
    }
```

P1 Reducer

문제 2 해결 과정

-MAP

- input: < Key, Value, context> Type: <Object, Text, Context>

key로 해당 파일을 읽어 들인 후 Value는 파일에서 한줄 씩 읽어온다.

공기질 종류의 코드가 8이나 9가 아니고, PM10과 PM 2.5 기준 공기질이 “좋음” 범위에 해당하지 않은 값들은 처리하지 않는다. 또한 해당 문제도 column을 갖고오지 않도록 진행한다. 이후 문제 3, 4 쪽 파트도 똑같이 진행 하였으니 해당 코드쪽 설명은 생략하겠다.

```java
    @Override
    protected void map(Object key, Text value, Context context) throws IOException, Interr

        StringTokenizer st = new StringTokenizer(value.toString(), ",");

        // set Key
//      String dateCode = st.nextToken();                                                   //
//      String stationCode = st.nextToken();    // 101
//      Integer itemCode = Integer.parseInt(st.nextToken());

        String date = st.nextToken(); // 2017-01
        if (date.startsWith("Measurement")) return;
        String stationCode = st.nextToken();    // 101
        if (stationCode.startsWith("Station")) return;

        Integer itemCode = Integer.parseInt(st.nextToken());            // 1

        // itemCode == 8 -> PM10, itemCode == 9 -> PM2.5
        if(itemCode != 8 && itemCode != 9) return;

        Double itemValue = Double.parseDouble(st.nextToken());  // 0.004
        switch(itemCode){
            case 8:
                if(!(itemValue <= 30 && itemValue >= 0)) return;
                break;
            case 9:
                if(!(itemValue <= 15 && itemValue >= 0)) return;
                break;
        }

        String statusCode = st.nextToken();     // 0
        if (statusCode.startsWith("Instrument")) return;
        if(Integer.parseInt(statusCode) != 0) return;

        ok.set(stationCode);

        ov.set(date);

        // emit
        context.write(ok, ov);
    }
```

P2 Mapper

문제 2 해결 과정

-Reduce

- input: < Key,values> Type: <Text, Iterable<Text>

reduce에서 묶은 값들을 연산과정을 통해 처리한다.

ArrayList를 사용하여 key값으로 들어오는 값들이 있으면 cnt를 증가시키고
없으면 추가하여 max값을 산출 한다.

```java
public class P2Reducer extends Reducer<Text, Text, Text, Text> {

    Text ok = new Text();
    Text ov = new Text();
    int max = Integer.MIN_VALUE;
    String station;

    @Override
    protected void reduce(Text key, Iterable<Text> values, Reducer<Text,Text,Text,Text> .Con
            throws IOException, InterruptedException {

        List<String> date_List = new ArrayList<String>();
        // 101

        int cnt = 0;
        for(Text val: values) {
            if(date_List.contains(val.toString())) {
                cnt ++;
            }
            else date_List.add(val.toString());
        }
        if(max<cnt) {
            max = cnt;
            station = key.toString();
        }

        ok.set("현재 state code: "+ key + ", count: "+ cnt);
        ov.set("최대값; " + station);

        // emit
        context.write(ok, ov);
    }
}
```

문제2 Reducer

문제 3 해결 과정

-MAP

- input: < Key, Value, context> Type: <Object, Text, Context>

  key로 해당 파일을 읽어 들인 후 Value는 파일에서 한줄 씩 읽어온다.

key값으로 시간 + 지역 코드 , value로 공기질 종류 코드 + 공기질 코드 값을 넘겨준다. 이때 statuscode가 0 이 아니면 Reduce로 넘기지 않는다.

```java
ublic class P3Mapper extends Mapper<Object, Text, Text, Text> {

    Text ok = new Text();
    Text ov = new Text();

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, Interr
        // 2017-01-01 00:00, 101, 1, 0.004, 0
        StringTokenizer st = new StringTokenizer(value.toString(), ",");

        String date = st.nextToken(); // 2017-01
        if (date.startsWith("Measurement")) return;
        String stationCode = st.nextToken();    // 101
        if (stationCode.startsWith("Station")) return;

        String itemCode = st.nextToken();           // 1
        if( itemCode.startsWith("Item")) return;

        // set Value
        Double itemValue = Double.parseDouble(st.nextToken());  // 0.004

        String statusCode = st.nextToken();     // 0
        if (statusCode.startsWith("Instrument")) return;

        if(Integer.parseInt(statusCode) != 0) return;   // statusCode == 0 -> normal data

        ok.set(date + "\t" + stationCode);

//      System.out.println(timeCode + "\t" + stationCode);

        ov.set(itemCode + ": " + itemValue.toString());

//      System.out.println(itemCode + ": " + itemValue.toString());

        // emit
        context.write(ok, ov);
    }
```

문제3 Mapper

문제 3 해결 과정

-Reduce

- input: < Key,values, context> Type: <Text, Iterable<Text>, context>

  reduce에서 묶은 값들을 연산과정을 통해 처리한다.

해당 공기질 코드를 case별로 값을 가져와 case문으로 조건을 걸어 시간,지역 별로 모든 종류의 측정치 값들을 가져와 진행한다.

```java
ublic class P3Reducer extends Reducer<Text, Text, Text, Text> {

    Text ov = new Text();

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOExcepti

        String ovString = "";
        String temp = "<";
        for(Text t : values){

            StringTokenizer st = new StringTokenizer(t.toString(), ":");
            String itemName = "";
            int itemCode = Integer.parseInt(st.nextToken());
            switch(itemCode){
                case 1:
                    itemName = "SO2";
                    break;
                case 3:
                    itemName = "NO2";
                    break;
                case 5:
                    itemName = "CO";
                    break;
                case 6:
                    itemName = "O3";
                    break;
                case 8:
                    itemName = "PM10";
                    break;
                case 9:
                    itemName = "PM2.5";
                    break;
            }
            String itemValue = st.nextToken();
            temp += (itemName + ": " + itemValue + ", ");
        }

        ovString = temp.substring(0, temp.length()-2);
        ovString += ">";

        System.out.println(ovString);
        ov.set(ovString);

        context.write(key, ov);
```

문제 4 해결 과정

-MAP

- input: < Key, Value, context> Type: <Object, Text, Context>

  key로 해당 파일을 읽어 들인 후 Value는 파일에서 한줄 씩 읽어온다.

key값으로 시간, value로 공기질 종류 코드 + 공기질 코드 값을 넘겨준다.
이때 statuscode가 0 이 아니면 Reduce로 넘기지 않는다.

```java
ublic class P3Mapper extends Mapper<Object, Text, Text, Text> {

    Text ok = new Text();
    Text ov = new Text();

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, Inter
        // 2017-01-01 00:00, 101, 1, 0.004, 0
        StringTokenizer st = new StringTokenizer(value.toString(), ",");

        String date = st.nextToken(); // 2017-01
        if (date.startsWith("Measurement")) return;
        String stationCode = st.nextToken();    // 101
        if (stationCode.startsWith("Station")) return;

        String itemCode = st.nextToken();           // 1
        if( itemCode.startsWith("Item")) return;

        // set Value
        Double itemValue = Double.parseDouble(st.nextToken());  // 0.004

        String statusCode = st.nextToken();     // 0
        if (statusCode.startsWith("Instrument")) return;

        if(Integer.parseInt(statusCode) != 0) return;   // statusCode == 0 -> normal data

        ok.set(date + "\t" + stationCode);

//      System.out.println(timeCode + "\t" + stationCode);

        ov.set(itemCode + ": " + itemValue.toString());

//      System.out.println(itemCode + ": " + itemValue.toString());

        // emit
        context.write(ok, ov);
    }
```

문제3 Mapper

문제 4 해결 과정

-Reduce

- input: < Key,values, context> Type: <Text, Iterable<Text>, context>

reduce에서 묶은 값들을 연산과정을 통해 처리한다.

시간대 별로 평균 공기질을 구해야 하기 때문에 case문을 통해 값들을 구한 후 이를 평균화 한다.

```
Text ok = new Text();
Text ov = new Text();

@Override
protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

    double sumSO2 = 0, sumNO2 = 0, sumCO = 0, sumO3 = 0, sumPM10 = 0, sumPM2d5 = 0;
    int cntSO2 = 0, cntNO2 = 0, cntCO = 0, cntO3 = 0, cntPM10 = 0, cntPM2d5 = 0;

    for(Text t : values){
        StringTokenizer st = new StringTokenizer(t.toString());

        Integer itemCode = Integer.parseInt(st.nextToken());
        Double itemValue = Double.parseDouble(st.nextToken());
        switch(itemCode){
            case 1:  // SO2
                sumSO2 += itemValue;
                cntSO2++;
                break;
            case 3:  //NO2
                sumNO2 += itemValue;
                cntNO2++;
                break;
            case 5: //CO
                sumCO += itemValue;
                cntCO++;
                break;
            case 6: //O3
                sumO3 += itemValue;
                cntO3++;
                break;
            case 8: //PM10
                sumPM10 += itemValue;
                cntPM10++;
                break;
            case 9: //PM2.5
                sumPM2d5 += itemValue;
                cntPM2d5++;
                break;
        }
    }
    Double avgSO2 = new Double(sumSO2 / cntSO2);
    Double avgNO2 = new Double(sumNO2 / cntNO2);
    Double avgCO = new Double(sumCO / cntCO);
    Double avgO3 = new Double(sumO3 / cntO3);
    Double avgPM10 = new Double(sumPM10 / cntPM10);
    Double avgPM2d5 = new Double(sumPM2d5 / cntPM2d5);

    String ovString = "<" + "SO2: " + avgSO2.toString() + ", NO2: " + avgNO2.toString() + ", CO: " + avgCO + ", O3: " + avgO3.to

    ov.set(ovString);

    context.write(key, ov);
}
```

```
101 8   37.733367431246805  289.0   3.0
102 8   38.04298686620679   296.0   3.0
103 8   35.90305508780371   330.0   3.0
104 8   42.10383539752297   423.0   1.0
105 8   42.61197038255862   401.0   3.0
106 8   43.92973977695167   389.0   3.0
107 8   44.3444359109099    411.0   3.0
108 8   41.40356203197733   340.0   3.0
109 8   39.77476783324547   326.0   3.0
110 8   39.2093618057811    414.0   3.0
111 8   44.38306386418755   421.0   3.0
112 8   39.002576615264495  322.0   2.0
113 8   40.84886350113055   354.0   1.0
114 8   40.53352042407234   289.0   3.0
115 8   42.79501330828245   293.0   3.0
116 8   43.93804243008679   389.0   3.0
117 8   43.38481526593585   405.0   3.0
118 8   39.6592073862525    329.0   3.0
119 8   47.119630083389325  351.0   3.0
120 8   41.863273727647865  321.0   3.0
121 8   44.95263464580038   385.0   3.0
122 8   44.45966335428919   470.0   1.0
123 8   39.86824337821102   302.0   1.0
124 8   42.44131683248403   426.0   1.0
125 8   45.14845245692406   443.0   1.0
```

Problem 1 결과 값

```
현재 state code: 101, count: 8778최대값; 101
현재 state code: 102, count: 8978최대값; 102
현재 state code: 103, count: 8884최대값; 102
현재 state code: 104, count: 7025최대값; 102
현재 state code: 105, count: 7836최대값; 102
현재 state code: 106, count: 6345최대값; 102
현재 state code: 107, count: 7411최대값; 102
현재 state code: 108, count: 7575최대값; 102
현재 state code: 109, count: 8101최대값; 102
현재 state code: 110, count: 8517최대값; 102
현재 state code: 111, count: 6929최대값; 102
현재 state code: 112, count: 9669최대값; 112
현재 state code: 113, count: 8768최대값; 112
현재 state code: 114, count: 7728최대값; 112
현재 state code: 115, count: 7333최대값; 112
현재 state code: 116, count: 7396최대값; 112
현재 state code: 117, count: 7728최대값; 112
현재 state code: 118, count: 7775최대값; 112
현재 state code: 119, count: 6069최대값; 112
현재 state code: 120, count: 7685최대값; 112
현재 state code: 121, count: 6398최대값; 112
현재 state code: 122, count: 7742최대값; 112
현재 state code: 123, count: 8425최대값; 112
현재 state code: 124, count: 7970최대값; 112
현재 state code: 125, count: 7020최대값; 112
```

Problem 2 결과 값

```
2017-01-01 00:00    101 <O3:  0.002, SO2:  0.004, CO:  1.2, NO2:  0.059000000000000004, PM2.5:
2017-01-01 00:00    102 <SO2:  0.006, NO2:  0.068, CO:  1.3, O3:  0.002, PM10:  77.0, PM2.5:
2017-01-01 00:00    103 <O3:  0.002, SO2:  0.005, NO2:  0.039, CO:  1.4, PM10:  70.0, PM2.5:
2017-01-01 00:00    104 <SO2:  0.005, NO2:  0.045, CO:  0.6, O3:  0.003, PM10:  73.0, PM2.5:
2017-01-01 00:00    105 <SO2:  0.005, NO2:  0.044000000000000004, CO:  1.0, O3:  0.004, PM10:
2017-01-01 00:00    106 <SO2:  0.005, NO2:  0.066, CO:  1.5, O3:  0.003, PM10:  71.0, PM2.5:
2017-01-01 00:00    107 <SO2:  0.005, NO2:  0.049, CO:  0.9, O3:  0.002, PM10:  64.0, PM2.5:
2017-01-01 00:00    108 <SO2:  0.004, NO2:  0.045, CO:  0.8, O3:  0.003, PM10:  68.0, PM2.5:
2017-01-01 00:00    109 <SO2:  0.006, NO2:  0.052000000000000005, CO:  1.1, O3:  0.002, PM10:
2017-01-01 00:00    110 <SO2:  0.005, NO2:  0.04, CO:  0.8, O3:  0.002, PM10:  91.0, PM2.5:  5
2017-01-01 00:00    111 <SO2:  0.005, NO2:  0.047, CO:  0.9, O3:  0.002, PM10:  62.0, PM2.5:
2017-01-01 00:00    112 <NO2:  0.046, CO:  1.2, O3:  0.001, SO2:  0.004, PM10:  63.0, PM2.5:
2017-01-01 00:00    113 <SO2:  0.006, NO2:  0.051, CO:  0.9, O3:  0.002, PM10:  81.0, PM2.5:
2017-01-01 00:00    114 <SO2:  0.008, NO2:  0.055, CO:  1.4, O3:  0.002, PM10:  75.0, PM2.5:
2017-01-01 00:00    115 <SO2:  0.005, NO2:  0.055, CO:  1.3, O3:  0.002, PM10:  75.0, PM2.5:
2017-01-01 00:00    116 <O3:  0.002, SO2:  0.006999999999999999, NO2:  0.07, CO:  1.3, PM10:
2017-01-01 00:00    117 <SO2:  0.006999999999999999, NO2:  0.045, CO:  1.3, O3:  0.003, PM10:
2017-01-01 00:00    118 <SO2:  0.004, NO2:  0.06, CO:  1.2, O3:  0.001, PM10:  67.0, PM2.5:  4
2017-01-01 00:00    119 <SO2:  0.005, NO2:  0.035, CO:  1.5, O3:  0.004, PM10:  70.0, PM2.5:
2017-01-01 00:00    120 <SO2:  0.006, NO2:  0.062, CO:  1.2, O3:  0.002, PM10:  63.0, PM2.5:
2017-01-01 00:00    121 <O3:  0.004, SO2:  0.006, NO2:  0.075, CO:  1.5, PM10:  75.0, PM2.5:
2017-01-01 00:00    122 <NO2:  0.039, CO:  1.3, O3:  0.005, SO2:  0.005, PM10:  82.0, PM2.5:
2017-01-01 00:00    123 <SO2:  0.005, NO2:  0.04, CO:  0.8, O3:  0.002, PM10:  63.0, PM2.5:  4
2017-01-01 00:00    124 <O3:  0.003, SO2:  0.006, NO2:  0.04, CO:  1.4, PM10:  60.0, PM2.5:  5
2017-01-01 00:00    125 <SO2:  0.004, NO2:  0.042, CO:  0.9, O3:  0.002, PM10:  68.0, PM2.5:
2017-01-01 01:00    101 <SO2:  0.004, NO2:  0.057999999999999996, CO:  1.2, O3:  0.002, PM10:
2017-01-01 01:00    102 <SO2:  0.006, NO2:  0.066, CO:  1.4, O3:  0.002, PM10:  76.0, PM2.5:
2017-01-01 01:00    103 <SO2:  0.004, NO2:  0.038, CO:  1.4, O3:  0.002, PM10:  73.0, PM2.5:
2017-01-01 01:00    104 <SO2:  0.005, NO2:  0.044000000000000004, CO:  0.6, O3:  0.003, PM10:
2017-01-01 01:00    105 <SO2:  0.005, NO2:  0.042, CO:  0.9, O3:  0.004, PM10:  67.0, PM2.5:
2017-01-01 01:00    106 <SO2:  0.004, NO2:  0.064, CO:  1.5, O3:  0.003, PM10:  70.0, PM2.5:
2017-01-01 01:00    107 <SO2:  0.004, NO2:  0.046, CO:  0.8, O3:  0.002, PM10:  54.0, PM2.5:
2017-01-01 01:00    108 <NO2:  0.045, CO:  0.9, O3:  0.003, SO2:  0.004, PM10:  71.0, PM2.5:
2017-01-01 01:00    109 <SO2:  0.005, NO2:  0.05, CO:  1.2, O3:  0.002, PM10:  62.0, PM2.5:  4
2017-01-01 01:00    110 <SO2:  0.005, NO2:  0.043, CO:  0.8, O3:  0.002, PM10:  75.0, PM2.5:
2017-01-01 01:00    111 <SO2:  0.005, NO2:  0.046, CO:  0.9, O3:  0.002, PM10:  61.0, PM2.5:
2017-01-01 01:00    112 <O3:  0.002, SO2:  0.005, NO2:  0.046, CO:  1.0, PM10:  60.0, PM2.5:
2017-01-01 01:00    113 <SO2:  0.006, NO2:  0.051, CO:  1.0, O3:  0.002, PM10:  71.0, PM2.5:
```

Problem 3 결과 값 일부

```
2017-01-01 00:00    <SO2: 0.005320000000000002, NO2: 0.05076000000000007, CO: 1.148, O3: 0.002
2017-01-01 01:00    <SO2: 0.005120000000000001, NO2: 0.04928000000000001, CO: 1.148, O3: 0.0025
2017-01-01 02:00    <SO2: 0.004880000000000002, NO2: 0.04752000000000001, CO: 1.132000000000000
2017-01-01 03:00    <SO2: 0.004680000000000002, NO2: 0.04444000000000001, CO: 1.084, O3: 0.0024
2017-01-01 04:00    <SO2: 0.004520000000000014, NO2: 0.04176000000000001, CO: 1.080000000000000
2017-01-01 05:00    <SO2: 0.004640000000000002, NO2: 0.04064000000000001, CO: 1.076, O3: 0.0023
2017-01-01 06:00    <SO2: 0.004560000000000016, NO2: 0.04064000000000001, CO: 1.1, O3: 0.00236
2017-01-01 07:00    <SO2: 0.004680000000000002, NO2: 0.03980000000000016, CO: 1.063999999999999
2017-01-01 08:00    <SO2: 0.004680000000000002, NO2: 0.04028, CO: 1.024, O3: 0.0026400000000000
2017-01-01 09:00    <SO2: 0.004760000000000002, NO2: 0.04124000000000006, CO: 1.052, O3: 0.002
2017-01-01 10:00    <SO2: 0.005040000000000002, NO2: 0.04188000000000001, CO: 1.000000000000000
2017-01-01 11:00    <SO2: 0.005280000000000003, NO2: 0.04316000000000004, CO: 0.98400000000000
2017-01-01 12:00    <SO2: 0.005680000000000002, NO2: 0.0444, CO: 0.9279999999999999, O3: 0.0063
2017-01-01 13:00    <SO2: 0.006080000000000002, NO2: 0.04656000000000002, CO: 0.868, O3: 0.0074
2017-01-01 14:00    <SO2: 0.006520000000000002, NO2: 0.04668000000000001, CO: 0.76, O3: 0.00952
2017-01-01 15:00    <SO2: 0.006560000000000025, NO2: 0.04724000000000001, CO: 0.763999999999999
2017-01-01 16:00    <SO2: 0.006240000000000025, NO2: 0.04916000000000001, CO: 0.775999999999999
2017-01-01 17:00    <SO2: 0.005920000000000002, NO2: 0.05152, CO: 0.8599999999999999, O3: 0.006
2017-01-01 18:00    <SO2: 0.005520000000000015, NO2: 0.053840000000000006, CO: 0.92, O3: 0.003
2017-01-01 19:00    <SO2: 0.005400000000000001, NO2: 0.05352000000000005, CO: 0.988, O3: 0.003
2017-01-01 20:00    <SO2: 0.005480000000000001, NO2: 0.05196000000000006, CO: 1.008, O3: 0.003
2017-01-01 21:00    <SO2: 0.005240000000000025, NO2: 0.05031999999999999, CO: 0.952000000000000
2017-01-01 22:00    <SO2: 0.005160000000000001, NO2: 0.04988000000000001, CO: 0.956000000000000
2017-01-01 23:00    <SO2: -0.03491999999999999, NO2: 0.00788, CO: 0.892, O3: -0.037399999999999
2017-01-02 00:00    <SO2: 0.005200000000000002, NO2: 0.04812000000000001, CO: 0.895999999999999
2017-01-02 01:00    <SO2: 0.005160000000000002, NO2: 0.04584000000000001, CO: 0.916, O3: 0.0030
2017-01-02 02:00    <SO2: 0.005120000000000002, NO2: 0.04468000000000001, CO: 0.883999999999999
2017-01-02 03:00    <SO2: 0.005160000000000001, NO2: 0.04296000000000001, CO: 0.888, O3: 0.0030
2017-01-02 04:00    <SO2: 0.005160000000000002, NO2: 0.04192000000000001, CO: 0.863999999999999
2017-01-02 05:00    <SO2: 0.005000000000000001, NO2: 0.04180000000000002, CO: 0.892, O3: 0.0032
2017-01-02 06:00    <SO2: 0.005080000000000002, NO2: 0.04420000000000001, CO: 0.935999999999999
2017-01-02 07:00    <SO2: 0.005120000000000002, NO2: 0.04632000000000014, CO: 0.932, O3: 0.002
```

Problem 4 결과 값 일부

```
pyeoungsunho@kmu-bigdata-m:~$ hadoop jar seoul_air.jar seoul.p1.Problem1 -Dmapre
duce.job.reduces=3 Measurement_info.csv
2022-06-10 07:34:10,762 INFO client.RMProxy: Connecting to ResourceManager at kmu-bigdata-m/10.178.0.2:8032
2022-06-10 07:34:10,999 INFO client.AHSProxy: Connecting to Application History server at kmu-bigdata-m/10.178.0.2:1
0200
2022-06-10 07:34:11,275 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/stag
ing/pyeoungsunho/.staging/job_1654844596770_0001
2022-06-10 07:34:12,033 INFO input.FileInputFormat: Total input files to process : 1
2022-06-10 07:34:12,035 WARN concurrent.ExecutorHelper: Thread (Thread[GetFileInfo #1,5,main]) interrupted:
java.lang.InterruptedException
        at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.java:510)
        at com.google.common.util.concurrent.FluentFuture$TrustedFuture.get(FluentFuture.java:88)
        at org.apache.hadoop.util.concurrent.ExecutorHelper.logThrowableFromAfterExecute(ExecutorHelper.java:48)
        at org.apache.hadoop.util.concurrent.HadoopThreadPoolExecutor.afterExecute(HadoopThreadPoolExecutor.java:90)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1157)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:750)
2022-06-10 07:34:12,182 INFO mapreduce.JobSubmitter: number of splits:1
2022-06-10 07:34:12,454 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1654844596770_0001
2022-06-10 07:34:12,457 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-10 07:34:12,704 INFO conf.Configuration: resource-types.xml not found
2022-06-10 07:34:12,704 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-06-10 07:34:13,177 INFO impl.YarnClientImpl: Submitted application application_1654844596770_0001
2022-06-10 07:34:13,262 INFO mapreduce.Job: The url to track the job: http://kmu-bigdata-m:8088/proxy/application_16
54844596770_0001/
2022-06-10 07:34:13,263 INFO mapreduce.Job: Running job: job_1654844596770_0001
```

```
pyeoungsunho@kmu-bigdata-m:~$ hadoop jar seoul_air.jar seoul.p2.Problem2 -Dmapreduce.job.reduces=3 Measurement_info.
csv
2022-06-10 07:35:24,026 INFO client.RMProxy: Connecting to ResourceManager at kmu-bigdata-m/10.178.0.2:8032
2022-06-10 07:35:24,323 INFO client.AHSProxy: Connecting to Application History server at kmu-bigdata-m/10.178.0.2:1
0200
2022-06-10 07:35:24,571 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/stag
ing/pyeoungsunho/.staging/job_1654844596770_0002
2022-06-10 07:35:25,222 INFO input.FileInputFormat: Total input files to process : 1
2022-06-10 07:35:25,224 WARN concurrent.ExecutorHelper: Thread (Thread[GetFileInfo #1,5,main]) interrupted:
java.lang.InterruptedException
        at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.java:510)
        at com.google.common.util.concurrent.FluentFuture$TrustedFuture.get(FluentFuture.java:88)
        at org.apache.hadoop.util.concurrent.ExecutorHelper.logThrowableFromAfterExecute(ExecutorHelper.java:48)
        at org.apache.hadoop.util.concurrent.HadoopThreadPoolExecutor.afterExecute(HadoopThreadPoolExecutor.java:90)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1157)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:750)
2022-06-10 07:35:25,357 INFO mapreduce.JobSubmitter: number of splits:1
2022-06-10 07:35:25,670 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1654844596770_0002
2022-06-10 07:35:25,672 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-10 07:35:25,993 INFO conf.Configuration: resource-types.xml not found
2022-06-10 07:35:25,994 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-06-10 07:35:26,130 INFO impl.YarnClientImpl: Submitted application application_1654844596770_0002
2022-06-10 07:35:26,256 INFO mapreduce.Job: The url to track the job: http://kmu-bigdata-m:8088/proxy/application_16
54844596770_0002/
2022-06-10 07:35:26,258 INFO mapreduce.Job: Running job: job_1654844596770_0002
```

예시) hadoop을 이용하여 각 Problem의 package와 class명을 기입한 후 Map-reduce작업을 진행합니다.

```
pyeoungsunho@kmu-bigdata-m:~$ hdfs dfs -ls Measurement_info.csv_p1.out/
Found 4 items
-rw-r--r--   2 pyeoungsunho hadoop          0 2022-06-10 07:34 Measurement_info.csv_p1.out/_SUCCE
SS
-rw-r--r--   2 pyeoungsunho hadoop        273 2022-06-10 07:34 Measurement_info.csv_p1.out/part-r
-00000
-rw-r--r--   2 pyeoungsunho hadoop        272 2022-06-10 07:34 Measurement_info.csv_p1.out/part-r
-00001
-rw-r--r--   2 pyeoungsunho hadoop        306 2022-06-10 07:34 Measurement_info.csv_p1.out/part-r
-00002
```

```
pyeoungsunho@kmu-bigdata-m:~$ hdfs dfs -ls Measurement_info.csv_p2.out/
Found 4 items
-rw-r--r--   2 pyeoungsunho hadoop          0 2022-06-10 08:45 Measurement_info.csv_p2.out/_SUCCE
SS
-rw-r--r--   2 pyeoungsunho hadoop        459 2022-06-10 08:45 Measurement_info.csv_p2.out/part-r
-00000
-rw-r--r--   2 pyeoungsunho hadoop        408 2022-06-10 08:44 Measurement_info.csv_p2.out/part-r
-00001
-rw-r--r--   2 pyeoungsunho hadoop        408 2022-06-10 08:45 Measurement_info.csv_p2.out/part-r
-00002
```

```
pyeoungsunho@kmu-bigdata-m:~$ hdfs dfs -ls Measurement_info.csv_p3.out/
Found 4 items
-rw-r--r--   2 pyeoungsunho hadoop          0 2022-06-10 07:44 Measurement_info.csv_p3.out/_SUCCE
SS
-rw-r--r--   2 pyeoungsunho hadoop   22574968 2022-06-10 07:44 Measurement_info.csv_p3.out/part-r
-00000
-rw-r--r--   2 pyeoungsunho hadoop   22569235 2022-06-10 07:43 Measurement_info.csv_p3.out/part-r
-00001
-rw-r--r--   2 pyeoungsunho hadoop   22572596 2022-06-10 07:44 Measurement_info.csv_p3.out/part-r
-00002
```

```
^Cpyeoungsunho@kmu-bigdata-m:~$ hdfs dfs -ls Measurement_info.csv_p4.out/
Found 4 items
-rw-r--r--   2 pyeoungsunho hadoop          0 2022-06-10 08:11 Measurement_info.csv_p4.out/_SUCCE
SS
-rw-r--r--   2 pyeoungsunho hadoop    1224276 2022-06-10 08:11 Measurement_info.csv_p4.out/part-r
-00000
-rw-r--r--   2 pyeoungsunho hadoop    1225117 2022-06-10 08:11 Measurement_info.csv_p4.out/part-r
-00001
-rw-r--r--   2 pyeoungsunho hadoop    1225447 2022-06-10 08:11 Measurement_info.csv_p4.out/part-r
-00002
```

이후 정상적으로 $> hdfs dfs -ls “결과물file”을 진행하면 결과값들이 나와있는것을 볼 수 있습니다. 앞서 reduce에 인자 값을 3으로 설정하여 file들이 3개로 나뉜 모습을 볼 수 있습니다.

```
pyeoungsunho@kmu-bigdata-m:~$ hdfs dfs -cat Measurement_info.csv_p1.out/part-r-00000
102     8       38.04298686620679       296.0   3.0
105     8       42.61197038255862       401.0   3.0
108     8       41.40356203197733       340.0   3.0
111     8       44.38306386418755       421.0   3.0
114     8       40.53352042407234       289.0   3.0
117     8       43.38481526593585       405.0   3.0
120     8       41.863273727647865      321.0   3.0
123     8       39.86824337821102       302.0   1.0
```

```
pyeoungsunho@kmu-bigdata-m:~$ hdfs dfs -cat Measurement_info.csv_p2.out/part-r-00000
현재 state code: 101, count: 8778       최대값; 101
현재 state code: 104, count: 7025       최대값; 101
현재 state code: 107, count: 7411       최대값; 101
현재 state code: 110, count: 8517       최대값; 101
현재 state code: 113, count: 8768       최대값; 101
현재 state code: 116, count: 7396       최대값; 101
현재 state code: 119, count: 6069       최대값; 101
현재 state code: 122, count: 7742       최대값; 101
현재 state code: 125, count: 7020       최대값; 101
```

```
2019-12-31 22:00        110     <PM10:  21.0, O3:  0.012, PM2.5:  10.0, NO2:  0.026000000000000002, CO:  0.4, SO2:  0.003>
2019-12-31 22:00        113     <SO2:  0.002, CO:  0.5, PM10:  17.0, O3:  0.013999999999999999, PM2.5:  14.0, NO2:  0.021>
2019-12-31 22:00        116     <SO2:  0.004, O3:  0.004, CO:  0.5, NO2:  0.037000000000000005, PM10:  29.0, PM2.5:  17.0>
2019-12-31 22:00        119     <CO:  0.6, O3:  0.005, SO2:  0.002, NO2:  0.033, PM10:  23.0, PM2.5:  14.0>
2019-12-31 22:00        122     <PM2.5:  13.0, PM10:  25.0, SO2:  0.003, O3:  0.013999999999999999, CO:  0.3, NO2:  0.037000000000000005>
2019-12-31 22:00        125     <O3:  0.004, PM2.5:  18.0, SO2:  0.003, PM10:  25.0, NO2:  0.04, CO:  0.5>
2019-12-31 23:00        102     <O3:  0.002, PM2.5:  18.0, PM10:  22.0, NO2:  0.036000000000000004, CO:  0.5, SO2:  0.003>
2019-12-31 23:00        105     <CO:  0.6, SO2:  0.003, O3:  0.005, PM10:  19.0, NO2:  0.036000000000000004, PM2.5:  10.0>
2019-12-31 23:00        108     <NO2:  0.03, PM2.5:  11.0, SO2:  0.002, O3:  0.004, CO:  0.5, PM10:  23.0>
2019-12-31 23:00        111     <O3:  0.003, SO2:  0.003, NO2:  0.045, PM2.5:  13.0, PM10:  27.0, CO:  0.8>
2019-12-31 23:00        114     <PM2.5:  18.0, SO2:  0.004, NO2:  0.03, PM10:  22.0, O3:  0.008, CO:  0.5>
2019-12-31 23:00        117     <PM10:  20.0, O3:  0.006, NO2:  0.038, PM2.5:  19.0, SO2:  0.004, CO:  0.4>
2019-12-31 23:00        120     <CO:  0.5, NO2:  0.04, PM2.5:  20.0, O3:  0.003, PM10:  28.0, SO2:  0.003>
2019-12-31 23:00        123     <CO:  0.5, SO2:  0.003, O3:  0.003, NO2:  0.039, PM2.5:  13.0, PM10:  19.0>
```

```
019-10-27 07:00         <SO2: 0.002800000000000001, NO2: 0.028040000000000013, CO: 0.5159999999999999, O3: 0.005080000000000001, PM10: 24.48, PM2.5: 13.88>
019-10-27 10:00         <SO2: 0.003000000000000014, NO2: 0.024320000000000012, CO: 0.4760000000000015, O3: 0.01780000000000001, PM10: 28.0, PM2.5: 15.16>
019-10-27 13:00         <SO2: 0.003160000000000013, NO2: 0.012240000000000006, CO: 0.3560000000000001, O3: 0.037080000000000016, PM10: 23.12, PM2.5: 11.28>
019-10-27 16:00         <SO2: 0.003120000000000012, NO2: 0.012400000000000005, CO: 0.3520000000000001, O3: 0.038960000000000015, PM10: 25.12, PM2.5: 13.56>
019-10-27 19:00         <SO2: 0.003640000000000001, NO2: 0.031080000000000014, CO: 0.5000000000000001, O3: 0.01984000000000001, PM10: 25.72, PM2.5: 15.04>
019-10-27 22:00         <SO2: 0.003200000000000002, NO2: 0.044400000000000016, CO: 0.6279999999999999, O3: 0.006200000000000003, PM10: 28.0, PM2.5: 17.2>
019-10-28 00:00         <SO2: 0.003080000000000016, NO2: 0.04404000000000001, CO: 0.6359999999999999, O3: 0.00408, PM10: 27.36, PM2.5: 17.36>
019-10-28 03:00         <SO2: 0.002920000000000016, NO2: 0.037080000000000016, CO: 0.6039999999999999, O3: 0.00384, PM10: 25.36, PM2.5: 16.16>
019-10-28 06:00         <SO2: 0.003040000000000015, NO2: 0.03504000000000001, CO: 0.584, O3: 0.03480000000000001, PM10: 26.76, PM2.5: 17.6>
019-10-28 09:00         <SO2: 0.003600000000000002, NO2: 0.035200000000000016, CO: 0.588, O3: 0.0
```

이후 $> hdfs dfs -cat 명령어를 통해 결과값을 확인 할 수 있습니다.
local에서 진행됐던 결과물과 동일한 것을 확인 할 수 있었습니다.

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources	Physical Mem Used %	Physical VCores Used %
5	0	0	5	0	<memory:0, vCores:0>	<memory:12288, vCores:4>	<memory:0, vCores:0>	23	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
2	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1, vCores:1>	<memory:6144, vCores:2>	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1654844596770_0007	pyeoungsunho	seoul_air.jar	MAPREDUCE	default	0	Fri Jun 10 17:44:20 +0900 2022	Fri Jun 10 17:44:20 +0900 2022	Fri Jun 10 17:45:15 +0900 2022	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1654844596770_0006	pyeoungsunho	seoul_air.jar	MAPREDUCE	default	0	Fri Jun 10 17:11:12 +0900 2022	Fri Jun 10 17:11:12 +0900 2022	Fri Jun 10 17:12:00 +0900 2022	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1654844596770_0003	pyeoungsunho	seoul_air.jar	MAPREDUCE	default	0	Fri Jun 10 16:43:14 +0900 2022	Fri Jun 10 16:43:14 +0900 2022	Fri Jun 10 16:44:05 +0900 2022	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1654844596770_0002	pyeoungsunho	seoul_air.jar	MAPREDUCE	default	0	Fri Jun 10 16:35:26 +0900 2022	Fri Jun 10 16:35:26 +0900 2022	Fri Jun 10 16:36:03 +0900 2022	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1654844596770_0001	pyeoungsunho	seoul_air.jar	MAPREDUCE	default	0	Fri Jun 10 16:34:12 +0900 2022	Fri Jun 10 16:34:14 +0900 2022	Fri Jun 10 16:34:55 +0900 2022	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0

Showing 1 to 5 of 5 entries First Previous 1 Next Last

이후 GCP에서 dataproc을 들어간 후 웹 인터페이스에 있는
YARN ResourceManager를 들어가면 Hadoop을 통한 분산 시스템 처리를
볼 수 있습니다. 중간에 p2 code를 잘못 보내서 현재 5개가 올라와있는것을
볼 수 있습니다.