CS236 Database Management Systems Project

Student Name: Ping He

Student ID: 861197283

My username: phe004

Node number on the Cluster: z3

1. Description of how you chose to separate the problem into different mapreduce jobs, with reasoning.

I plan to separate the problem into three mapreduce programs or jobs (their names are “MPJoin”,”PerMonth”,”PerState” respectively), and also include the pre-process of input data of the first mapreduce jobs and post-process of output data of the third mapreduce jobs.

Pre-process of input data includes the pre-process of location dataset and recordings dataset. Pre-process of location dataset is to retain data lines that “CTRY” field value is “US” and has “STATE” field value. Pre-process of recordings dataset is to retain the column “STN---”,”WBAN”,”YEARMODA”,”TEMP”, the other fields will not be used in this project, so we ignore those and delete those unused columns.

First mapreduce program “MPJoin” is to join the data from location dataset, recordings dataset. Because joining two dataset will help to find the corresponding state of the stations, then make it possible to group the stations by state in US.

Second mapreduce program “PerMonth” is to compute the average temperature of each station in each month. Because this will make it convenient to compute the average temperature of each state in each month.

Third mapreduce program “PerState” is to compute the average temperature of each state in each month, the average temperature and name of the highest month, average temperature and name of the lowest month, and difference between the two.

Post-process is to process the output of third mapreduce program, sort by the difference, and output the result as the form the project required.

2. Description of each mapreduce job

First mapreduce job ”MPJoin”

In the Map function, the input files are from two files location dataset and recordings dataset (these inputs are all after preprocess), I use the flag to records where data comes from, set flag ’0’ when the data line comes from recordings dataset, set flag ’1’ when the data line comes from location dataset. Output these recordings dataset with the key”STN---”, Output these location dataset with the key “USAF”.

In the Reduce function, Separate the input data lines by flag, and store in two arraylists, then I have known in the reduce function, the data lines from location dataset and recordings dataset which have the same “STN---” and “USAF” value. Thus we join the two arraylists, and output the join result with the key from the Map function output.

I name the output of first mapreduce job “MPJoin” is “result_join.txt”.

Estimate of runtime: around 11 seconds

Second mapreduce job “PerMonth”

In the Map function, the input file is the output of first mapreduce job “MPJoin” is “result_join.txt”. Output these data lines with the key ”STN---”.

In the Reduce function, separate the value part of the output of Map function, process the ”YEARMODA” field value, add the “TEMP” field value to corresponding month collection by its ”YEARMODA” field value, then calculate the average temperature of each month of each station. Output the average temperature of each month of each station with the key ”STN---”.

I name the output of second mapreduce job “PerMonth” is “result_permonth.txt”.

Estimate of runtime:around 9 seconds

Third mapreduce job “PerState”

In the Map function, the input file is the output of second mapreduce job “PerMonth” is “result_permonth.txt”. Output these data lines with the key ”STATE”.

In the Reduce function, I have known the data lines in the reduce function all have the same “STATE” field value, so separate the value part of the output of Map function, calculate the average temperature of each month of each state, then sort these and calculate the difference. Output these result with the key “STATE”

I name the output of Third mapreduce job “PerState” is “result_perstate.txt”.

Estimate of runtime: around 6 seconds

3. Description of how you chose to do the join

Join location dataset and recordings dataset by the value of “STN---” field in recordings dataset equals the value of “USAF” field in location dataset. Join method is **Reduce-side Join**.

4. Description of anything you did extra for the project

Combiner

About Combiner, Combiner can be used in the Second and Third Mapreduce job to collect the temperature in the same month and sum these, then the result will be used in calculate the average temperature. In this way, the data transmission between Map and Reduce will be decreased.

Different ways to do the Join

Map-side Join can be used in the situation that one input file (e.g. the recordings dataset in this project) is very large and one input file (e.g. the location dataset in this project) is very small. The process is to put the small input file in DistributedCache when submit the job, then split the small input file into lines and marked join key / value , put these from DistributedCache into internal memory, after that, scan the large input file to find the line with the same join key and join them.

Process the station in US but don’t have the state tag

Calculate the average temperature in each month in each year in each station, then we can cluster the station with close average temperature in each month even each date. The close in temperature data show the close in location of the stations.

5. How to run the Program

The reason to calculate the average temperature of each state in each month in each year instead of combing the four years recordings data together then calculating the average temperature of each state in each month: As we know, each year recording data contains different number of records, if calculating after join four years recordings, then the weight of year with larger number of recordings will show dominate in the final result. However, this's not reasonable, I think I should calculate the average temperature of each state in each month in each year then calculate the average for four years based on this. In this way, the weight of each year will be the same in the final result.

First to pre-process the data

The pre-process source code is in the folder "preprocess".

The process2006.java is the pre-process for the file 2006.txt. Remember to change the path parameters in the program file to the path of 2006.txt in your computer and the output path of u2006_process.txt. Because the pre-process for four years recordings is the same, so you need to change the path parameters according to the years file you want to process as processing the 2006.txt.

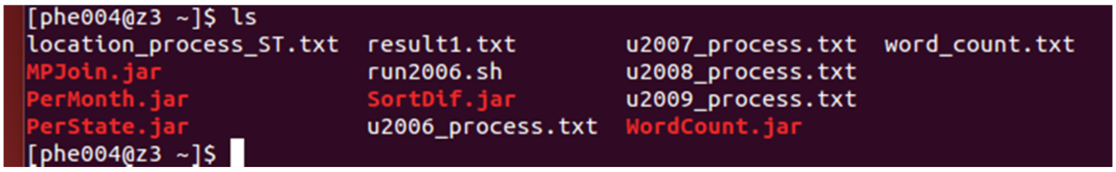
The readcvs.java is the pre-process for the file WeatherStationLocations.csv. Remember to change the path parameters in the program file to the path of WeatherStationLocations.csv in your computer and output path of location_process_ST.txt.

The pre-process result files are included in this project zip file.

Run the script

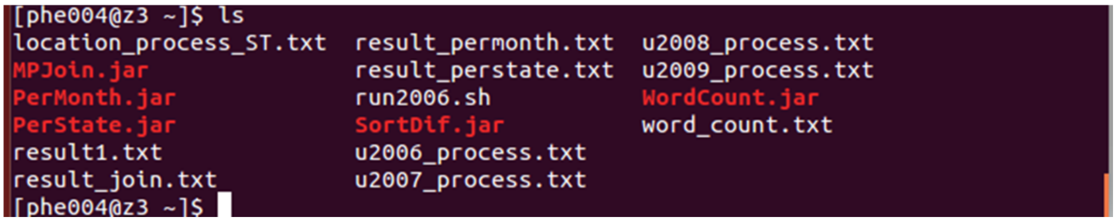
After Pre-process, the preprocess of each year data is in the corresponding year folder. E.g. the pre-process result of 2006.txt is the u2006_process.txt in the 2006 folder. The pre-process result of 2007.txt is the u2007_process.txt in the 2007 folder…… the pre-process result of WeatherStationLocations.csv is location_process_ST.txt.

Then put all files in "jar&sh" folder and the above noticed after preprocess files to the hadoop cluster.


```
[phe004@z3 ~]$ ls
location_process_ST.txt  result1.txt        u2007_process.txt  word_count.txt
MPJoin.jar               run2006.sh         u2008_process.txt
PerMonth.jar             SortDif.jar        u2009_process.txt
PerState.jar             u2006_process.txt  WordCount.jar
[phe004@z3 ~]$ 
```

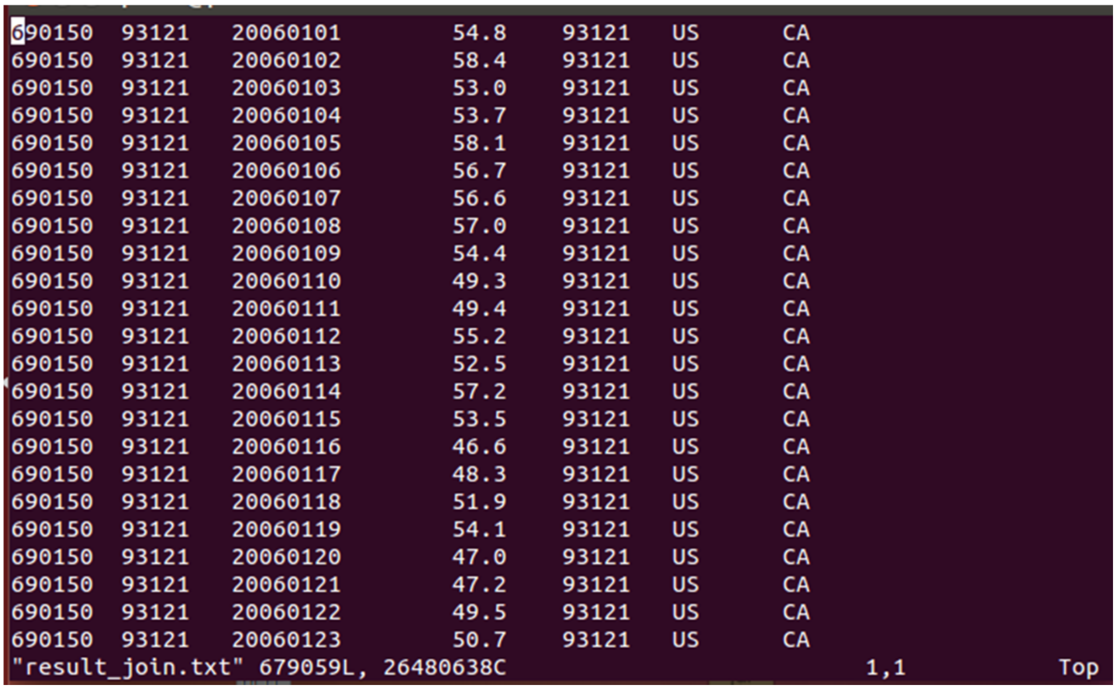
Before run the script, remember to delete all files and output folder in HDFS, because the duplication of file name will cause error.

Run the script by command line "sh run2006.sh"(this script is to process the data in 2006), then we can get three new files.



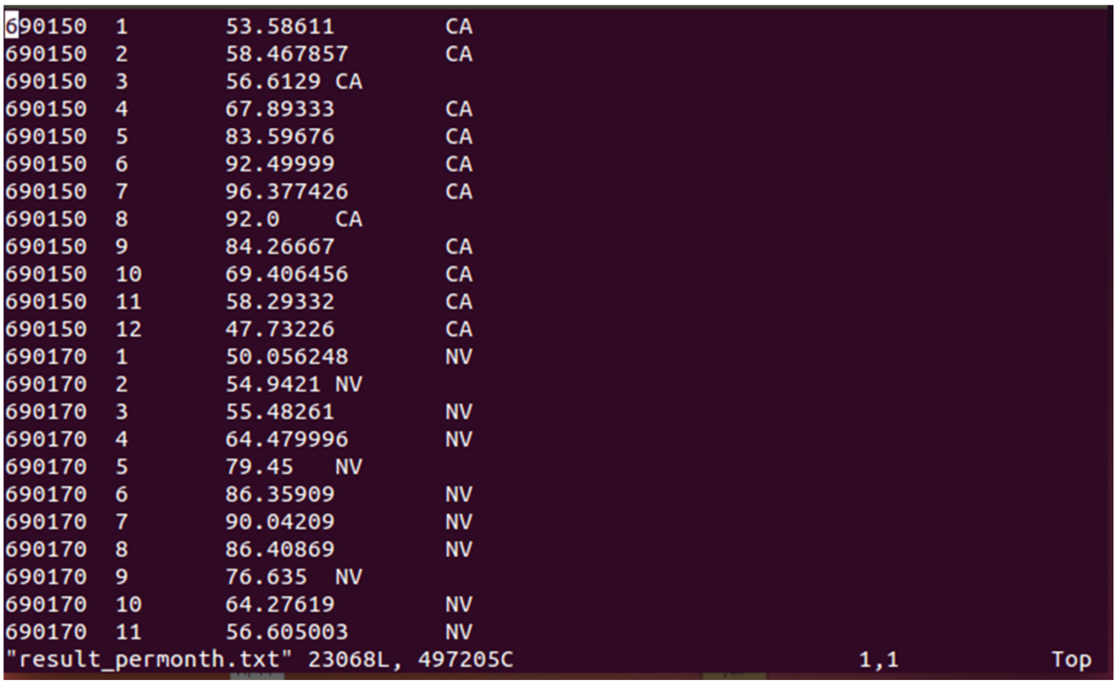
```
[phe004@z3 ~]$ ls
location_process_ST.txt  result_permonth.txt  u2008_process.txt
MPJoin.jar               result_perstate.txt  u2009_process.txt
PerMonth.jar             run2006.sh           WordCount.jar
PerState.jar             SortDif.jar          word_count.txt
result1.txt              u2006_process.txt
result_join.txt          u2007_process.txt
[phe004@z3 ~]$ 
```

The result_join.txt is the output of first job "MPJoin"



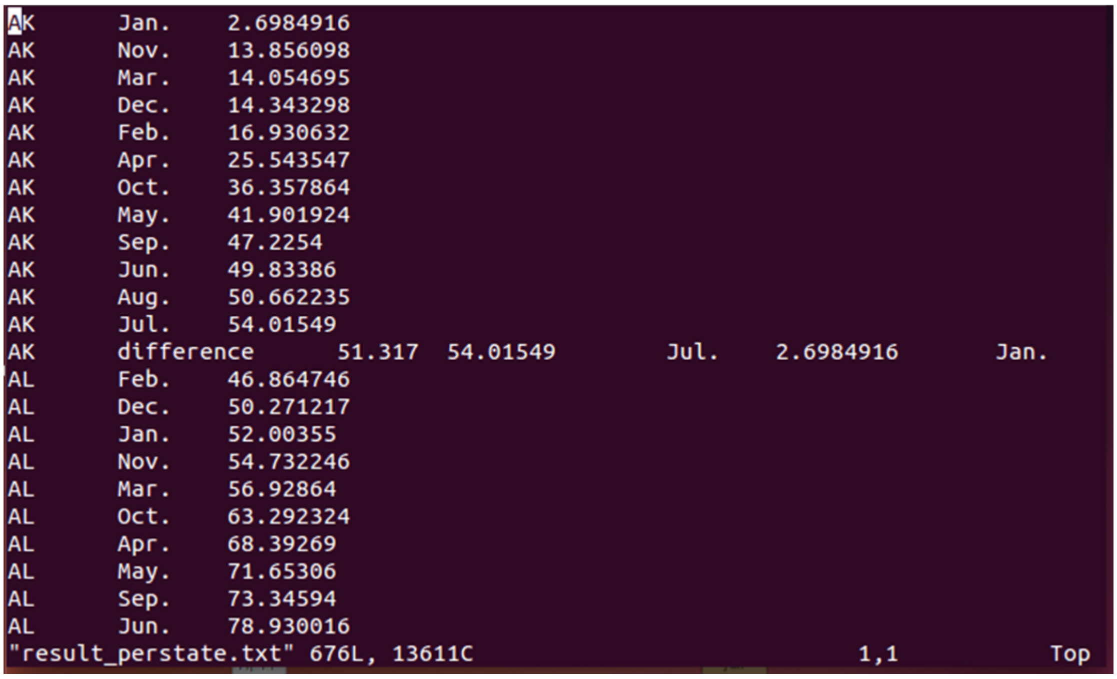
```
690150  93121   20060101        54.8    93121   US      CA
690150  93121   20060102        58.4    93121   US      CA
690150  93121   20060103        53.0    93121   US      CA
690150  93121   20060104        53.7    93121   US      CA
690150  93121   20060105        58.1    93121   US      CA
690150  93121   20060106        56.7    93121   US      CA
690150  93121   20060107        56.6    93121   US      CA
690150  93121   20060108        57.0    93121   US      CA
690150  93121   20060109        54.4    93121   US      CA
690150  93121   20060110        49.3    93121   US      CA
690150  93121   20060111        49.4    93121   US      CA
690150  93121   20060112        55.2    93121   US      CA
690150  93121   20060113        52.5    93121   US      CA
690150  93121   20060114        57.2    93121   US      CA
690150  93121   20060115        53.5    93121   US      CA
690150  93121   20060116        46.6    93121   US      CA
690150  93121   20060117        48.3    93121   US      CA
690150  93121   20060118        51.9    93121   US      CA
690150  93121   20060119        54.1    93121   US      CA
690150  93121   20060120        47.0    93121   US      CA
690150  93121   20060121        47.2    93121   US      CA
690150  93121   20060122        49.5    93121   US      CA
690150  93121   20060123        50.7    93121   US      CA
"result_join.txt" 679059L, 26480638C                 1,1           Top
```

The result_permonth.txt is the output of second job "PerMonth"



```
690150  1       53.58611        CA
690150  2       58.467857       CA
690150  3       56.6129 CA
690150  4       67.89333        CA
690150  5       83.59676        CA
690150  6       92.49999        CA
690150  7       96.377426       CA
690150  8       92.0    CA
690150  9       84.26667        CA
690150  10      69.406456       CA
690150  11      58.29332        CA
690150  12      47.73226        CA
690170  1       50.056248       NV
690170  2       54.9421 NV
690170  3       55.48261        NV
690170  4       64.479996       NV
690170  5       79.45   NV
690170  6       86.35909        NV
690170  7       90.04209        NV
690170  8       86.40869        NV
690170  9       76.635  NV
690170  10      64.27619        NV
690170  11      56.605003       NV
"result_permonth.txt" 23068L, 497205C                           1,1           Top
```

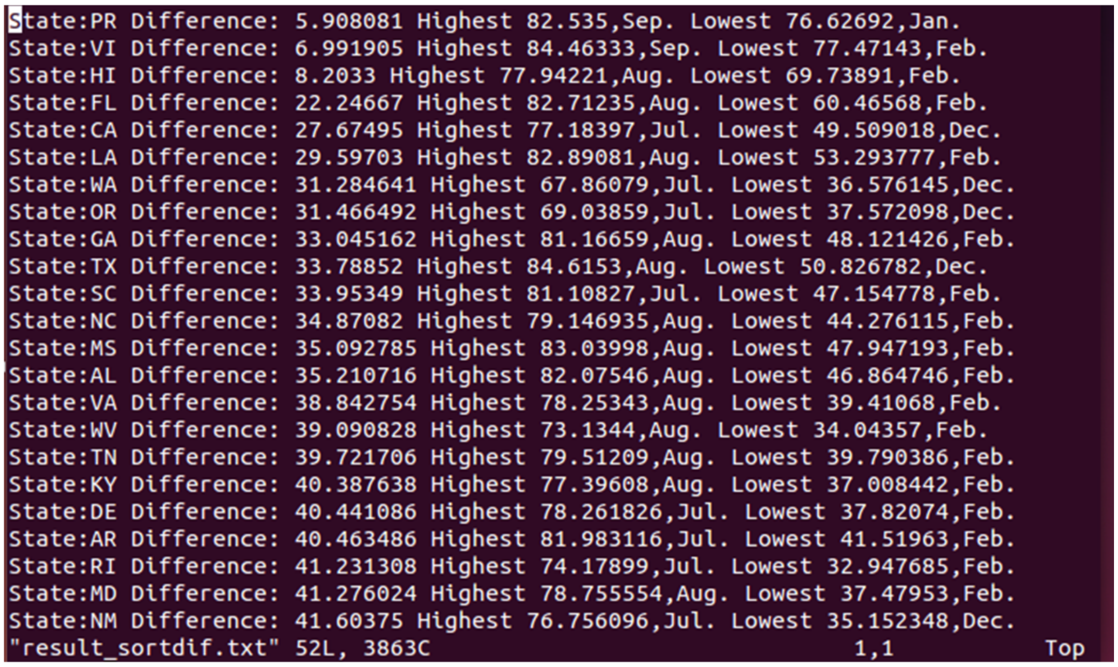
The result_perstate.txt is the output of third job “PerState”



```
AK      Jan.    2.6984916
AK      Nov.    13.856098
AK      Mar.    14.054695
AK      Dec.    14.343298
AK      Feb.    16.930632
AK      Apr.    25.543547
AK      Oct.    36.357864
AK      May.    41.901924
AK      Sep.    47.2254
AK      Jun.    49.83386
AK      Aug.    50.662235
AK      Jul.    54.01549
AK      difference      51.317  54.01549        Jul.    2.6984916       Jan.
AL      Feb.    46.864746
AL      Dec.    50.271217
AL      Jan.    52.00355
AL      Nov.    54.732246
AL      Mar.    56.92864
AL      Oct.    63.292324
AL      Apr.    68.39269
AL      May.    71.65306
AL      Sep.    73.34594
AL      Jun.    78.930016
"result_perstate.txt" 676L, 13611C                              1,1           Top
```

Post-process of the output third job

The output of third job result_perstate.txt is very small so we run the program “SortDif.java” in the “postprocess” folder. Remember to change the path parameters in the program file to the path of result_perstate.txt in your computer and output file result_sortdif.txt (this file must be created first) when you run the post-process program.



```
State:PR Difference: 5.908081 Highest 82.535,Sep. Lowest 76.62692,Jan.
State:VI Difference: 6.991905 Highest 84.46333,Sep. Lowest 77.47143,Feb.
State:HI Difference: 8.2033 Highest 77.94221,Aug. Lowest 69.73891,Feb.
State:FL Difference: 22.24667 Highest 82.71235,Aug. Lowest 60.46568,Feb.
State:CA Difference: 27.67495 Highest 77.18397,Jul. Lowest 49.509018,Dec.
State:LA Difference: 29.59703 Highest 82.89081,Aug. Lowest 53.293777,Feb.
State:WA Difference: 31.284641 Highest 67.86079,Jul. Lowest 36.576145,Dec.
State:OR Difference: 31.466492 Highest 69.03859,Jul. Lowest 37.572098,Dec.
State:GA Difference: 33.045162 Highest 81.16659,Aug. Lowest 48.121426,Feb.
State:TX Difference: 33.78852 Highest 84.6153,Aug. Lowest 50.826782,Dec.
State:SC Difference: 33.95349 Highest 81.10827,Jul. Lowest 47.154778,Feb.
State:NC Difference: 34.87082 Highest 79.146935,Aug. Lowest 44.276115,Feb.
State:MS Difference: 35.092785 Highest 83.03998,Aug. Lowest 47.947193,Feb.
State:AL Difference: 35.210716 Highest 82.07546,Aug. Lowest 46.864746,Feb.
State:VA Difference: 38.842754 Highest 78.25343,Aug. Lowest 39.41068,Feb.
State:WV Difference: 39.090828 Highest 73.1344,Aug. Lowest 34.04357,Feb.
State:TN Difference: 39.721706 Highest 79.51209,Aug. Lowest 39.790386,Feb.
State:KY Difference: 40.387638 Highest 77.39608,Aug. Lowest 37.008442,Feb.
State:DE Difference: 40.441086 Highest 78.261826,Jul. Lowest 37.82074,Feb.
State:AR Difference: 40.463486 Highest 81.983116,Jul. Lowest 41.51963,Feb.
State:RI Difference: 41.231308 Highest 74.17899,Jul. Lowest 32.947685,Feb.
State:MD Difference: 41.276024 Highest 78.755554,Aug. Lowest 37.47953,Feb.
State:NM Difference: 41.60375 Highest 76.756096,Jul. Lowest 35.152348,Dec.
"result_sortdif.txt" 52L, 3863C                              1,1           Top
```

The above description is to process the data in u2006_process.txt. You can use the same method to run the corresponding script in that year, e.g. run2007.sh is the script for process data in u2007_process.txt. Before run the script, remember to delete all files and output folder in HDFS, because the duplication of file name will cause error.

6. The result of the whole project and Conclusion

I have collected the formal result of each year:
result_sortdif.txt in the “2006” folder is the formal result of 2006.
result_sortdif.txt in the “2007” folder is the formal result of 2007.
result_sortdif.txt in the “2008” folder is the formal result of 2008.
result_sortdif.txt in the “2009” folder is the formal result of 2009.

These four years result files and the monthly average data computed in above are so small, so it’s easy to analyze and calculate the average temperature and difference in 4 years in total. Then we can get the first three states with least difference in 4 years in total:

State	Difference	Highest Month	Highest Temp.	Lowest Month	Lowest Temp.
VI	5.732	July	83.323	February	77.591
PR	6.009	August	82.635	January	76.626
HI	7.754	August	77.692	February	69.938

Conclusion: the State “VI” in US has the most stable temperature.