Data 420 Assignment – Meng Gao

Introduction

In this assignment we analyze a dataset called Global Historical Climatology Network (GHCN), which is an integrated database of climate summaries from land surface stations across the globe that have been subjected to a common suite of quality assurance reviews (National Climatic Data Center).

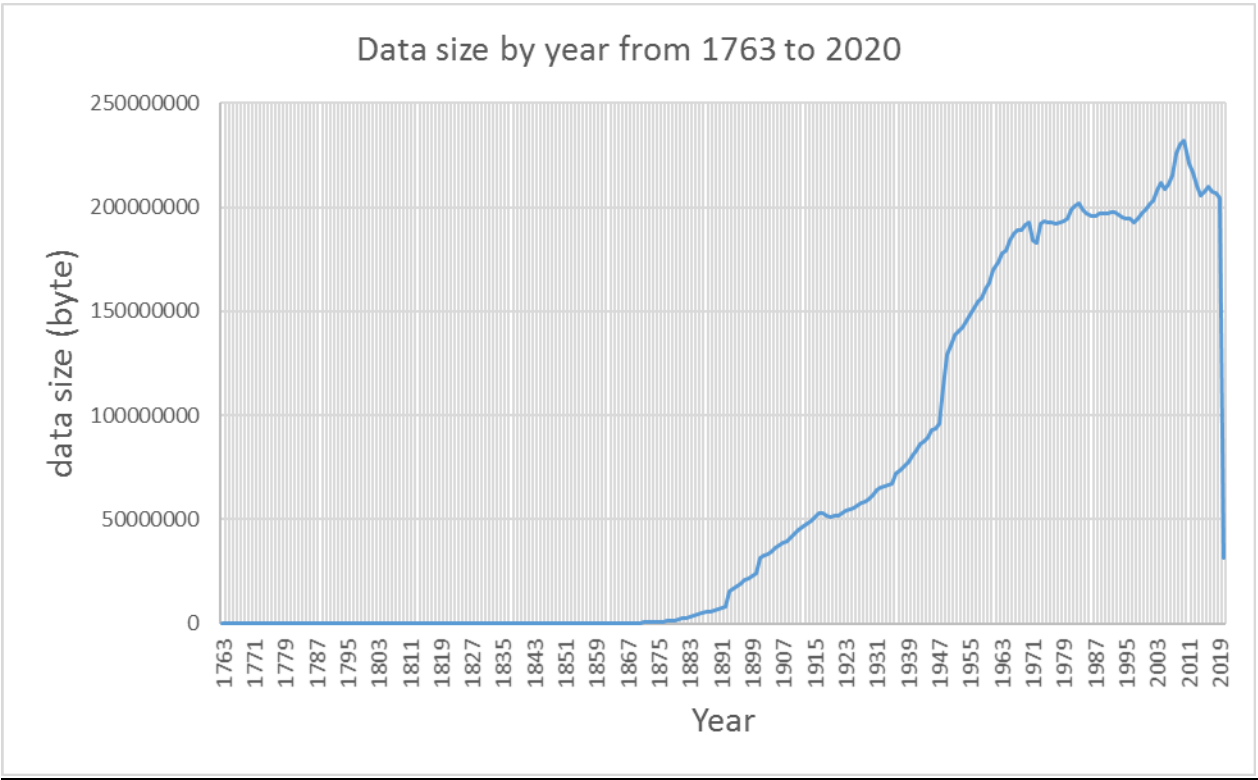
Processing

Q1 (a)The data is structured like the tree directory as shown below

```
 |---countries
 |---daily
 |-----1763.csv.gz
  …
 |-----2020.csv.gz
 |---inventory
 |---states
 |---stations
```

As we can see, there are four files named countries, inventory, states and stations. Daily is a working directory that contains 258 files from 1763.csv.gz to 2020.csv.gz, while the other three are single files.

(b) There are 258 files contained in daily directory, ranging from 1763.csv.gz to 2020.csv.gz.



The size of all data is about 15.535GB and the vast majority of it is daily data, which is about 15.496GB. The rest of the data is about 40MB. As the daily data is compressed csv and the compressed ratio is in the range from 10:1 to 100:1, the original file for daily is about 150GB to 1.5TB.

Q2. Having the first 1000 rows of daily loaded in pyspark and checked, I reckon description of the data was accurate. But the DATA column should be converted to DateType and TimeStampType after being loaded as they represent either a date or a time, we shouldn't simply treat them as strings.

After each file with the fixed width format is loaded into spark, the schemas we defined before do not seem to work. Fortunately, spark has different functions that allow us to split the fixed text value. trim(str) helps remove the leading and trailing space characters from str, and substring allows us to slice the string based on the start or end of length we choose, we can also define the datatypes based on the column names.

The number of rows in each metadata table can be seen in the table below.

Metadata table	stations	countries	states	Inventory
Number of rows	115081	219	74	687141

There are 106993 stations that do not have a WMO ID.

Q3. Spark's substring allows us to exact the two character country code and store it in a new column named 'COUNTRY-CODE'. The following step was to left join this station dataframe with countries and states. Based on inventory, we can compute aggregates with the minimum value of 'FIRSTYEAR' and the maximum value of 'LASTYEAR' to get a new table containing each station and its time period of being active. By counting the distinct rows in this table, we can also know that overall there are 115072 stations that collected elements.
By filtering the core elements in the inventory dataframe and counting rows, we know that 318583 is the total number of core elements each station has collected, while the number for 'other' elements is 368558.
A table that contains each station and the number of core elements or other elements is created once we group the dataframe containing only core elements by ELEMENT (see below).

ELEMENT	SNOW	SNWD	PRCP	TMIN	TMAX
count	66536	58844	113070	40013	40120

There are 20266 stations that have collected five core elements, to get this answer, what l did was to join the inventory dataframe containing only core elements with the stations dataframe containing elements and get the count of core elements in each station.
To know the number of stations that only collects collecting temperature element, l think we need to count the rows of stations that only have one core element, and it should be either TMAX' or 'TMIN'.
Since Parquet performs better compression and has a good read performance (Google Books), we are storing the outputs with parquet format.
To check if there any stations in the first 1000 rows of daily that are not in stations, we can simply filter 'STATION_NAME' column to see if there are any null values in it after left joining the subset of daily with the enriched stations data. Or we can do a simple calculation to see if there's any difference in the distinct ID values in the subset of daily and the IDs of the enriched stations data.

Analysis
Q1. By counting the rows in daily dataset, we know that there are 115081 stations in total, among which 32850 have been active in 2020. To get this output, what l did was count the rows with 'first' smaller than 2020 and 'last' bigger than it. By filtering different flags, we know 991,1218 and 233 stations are in the GCOS Surface Network (GSN), the US Historical Climatology Network (HCN), and the US Climate Reference Network (CRN) separately. There are at least 14 stations that are in more

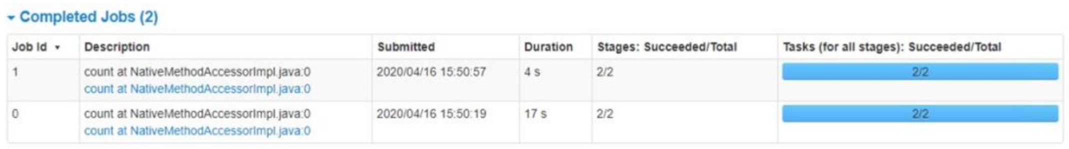
than one of these networks as when I filtered both HCN and CRN and counted the rows, the output was 14.
We can count the number of stations in each country or state by grouping by COUNTRY_CODE or. STATE_CODE, both of the outputs are saved into outputs directory.
By filtering the latitude bigger than zero and count the rows, we know that 89745 stations are in the Northern Hemisphere only. With the help of spark's 'like' function, we get the output that United states have 10 different territories including united states itself, 62183 is the number of stations in all the territories when we calculate the sum of all of the stations.

Q2. Using the function of commutating the geographical distance between two stations and having the output sorted in a descending order, we can know that NZ000933090 and NZ000093844 are the geographically furthest apart in New Zealand, with a distance of about 950.93km.

Q3. The default block size of HDFS is 134217728 B， which is about 128MB. One block is required to for the daily climate summaries for the year 2020, while the year 2010 data requires 2 blocks. The individual block sizes are displayed in the table below.

	HDFS	year 2010	year 2020
Default/individual block size	134217728 B	232080599 B	31626590B
Number of blocks	-	2	1

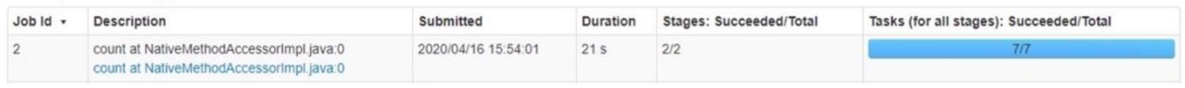
The screenshot is taken from my application console, which displays how many tasks are executed for daily summaries of year 2015 and 2020.
As can be seen in the table below, the number of tasks executed does not always correspond to the number of blocks in each input. The number of tasks is proportional to how many cores are available at the time, and one block can comprise of several nodes and executing several tasks.


Completed Jobs (2)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2020/04/16 15:50:57	4 s	2/2	2/2
0	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2020/04/16 15:50:19	17 s	2/2	2/2

	year 2015	year 2020
number of observations	34899013	5215364
Number of blocks	2	1
Number of tasks	2	2

When we read the files ranging from 2015.csv to 2020.csv into one single dataframe and count the rows, we can know that the number of observations in daily from 2015 to 2020 is 178918901.



Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2020/04/16 15:54:01	21 s	2/2	7/7

As can be seen in the screenshot above, 7 tasks were executed by each stage, which does not this correspond to your input as there're more than 7 blocks in daily from 2015 to 2020.

When one single file is loaded into spark, it creates one partition. When serval files are loaded and input files that are gzip compressed, as they cannot be splitted, the number of partitions is determined by the number of files and the file size.
When loading and applying transformations to daily, we can have at most 8 tasks, as we have 8 executors.
One way to increase the level of parallelism is to increase the number of executors. As we know, Spark automatically sets the number of “map” tasks to run on each file according to its size, so what we can do is control it by loading with different formats. Applying groupByKey or reduceByKey is also a good choice as it helps save memory. Or we can change the default shuffle partition.

Q4. There are 2928664265 rows in daily dataset. Having the frequency of all core elements counted, we know ‘PRCP’ has the most observations with a number of 1021682095.
8428939 observations of TMIN do not have a corresponding observation of TMAX. 27527 different stations contributed to these observations. The table below shows how many different core elements were collected. As we can see, ‘PRCP’ has the most observations, which is consistent with what we got in Processing 3.

ELEMENT	count
SNOW	332430532
SNWD	283572167
PRCP	1021682095
TMIN	435296249
TMAX	436709207

One simple way to calculate the number of observations of TMIN do not have a corresponding observation of TMAX is to create a table with both TMIN and TMAX, and then filter the observations that only TMIN and do not contain TMAX. By doing this, we know that there are 458892 observations of TMIN and TMAX for all stations in New Zealand, 81 years are covered by the observations and 27527 observations of TMIN do not have a corresponding observation of TMAX.

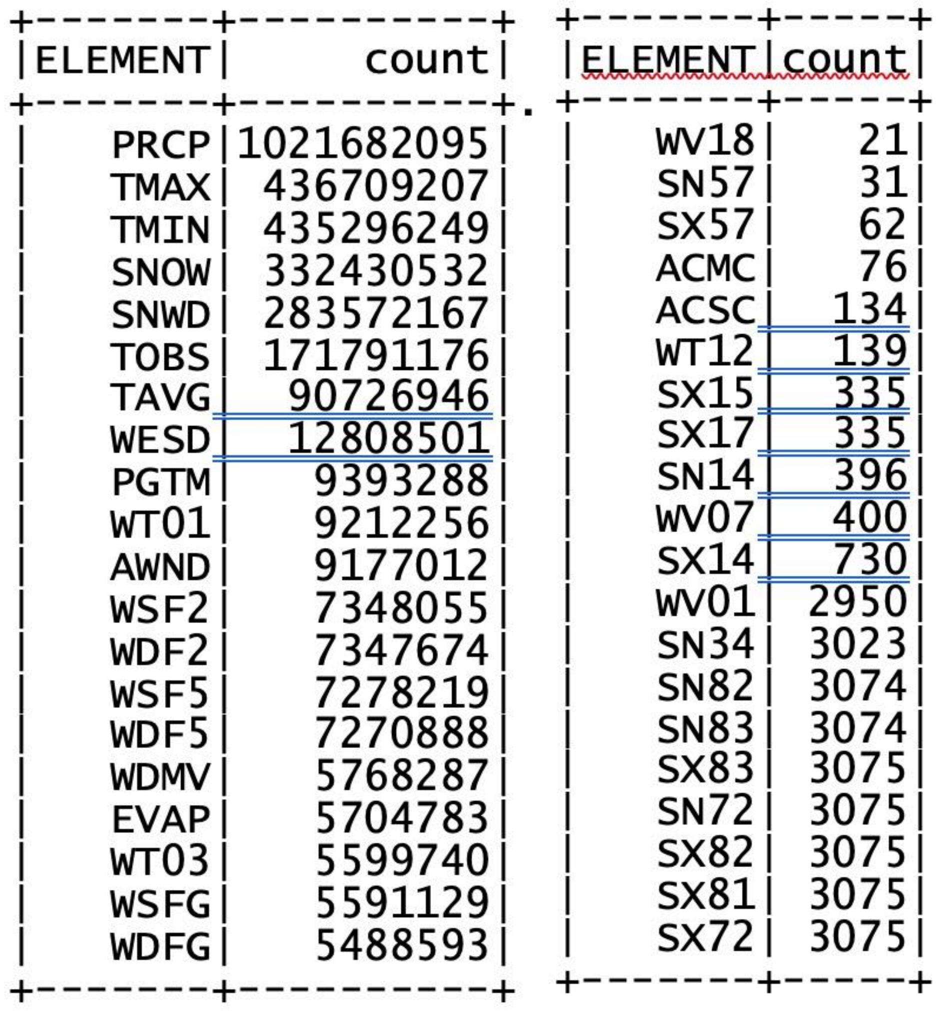
This time l saved the temperature_NZ with csv format, as the output is not of big size and it will be easier to load into R or Python. Before we actually do this, to merge all the part files into one single file makes it easier to count the rows and load into another language. When we check the csv file itself, we can know it has 458892 rows, which is exactly the number of observations that we counted using Spark before.

In R, l aggregated the temperature values by year and plotted the average time series for TMIN and TMAX for the entire country. but it wasn’t easy to plot for each station and the dual plot was not so easy to read, so l switched to Python, which allows me to set a MultiIndex of each station and date and parse the dates and replotted the average time series for TMIN and TMAX, as it would be easier to read.

Having a new table containing daily data and only precipitation created and grouped by country code, we can know in this entire dataset, Equatorial Guinea（country code EK） has the highest average rainfall in year 2000 with a number of 4361, which is 436.1mm. I would say there’s no evidence to show this number is not sensible, because according to CLIMATE-DATA, the rainfall there is around 2190 mm per year. Having a careful check in the dataset, we find out something unexpected, in the year of 2000, Equatorial Guinea only collected the precipitation once, and there are only two climate stations in the entire country. As it does rain a lot in Equatorial Guinea, it is possible that a precipitation of 436.1mm was collected on one of the rainy days there.

Challenges

In the daily dataset, when we group the daily data by ELEMENT and count the frequency of each element. We can get an output like below (the first 20 rows in ascending or descending order). With the core elements being the majority of the elements that have been collected, the other element covers 14% (which is 2928664265-2509690250)/ 2928664265) of all the elements.



```
+-------+----------+  +-------+-----+
|ELEMENT|     count|  |ELEMENT|count|
+-------+----------+. +-------+-----+
|   PRCP|1021682095|  |   WV18|   21|
|   TMAX| 436709207|  |   SN57|   31|
|   TMIN| 435296249|  |   SX57|   62|
|   SNOW| 332430532|  |   ACMC|   76|
|   SNWD| 283572167|  |   ACSC|  134|
|   TOBS| 171791176|  |   WT12|  139|
|   TAVG|  90726946|  |   SX15|  335|
|   WESD|  12808501|  |   SX17|  335|
|   PGTM|   9393288|  |   SN14|  396|
|   WT01|   9212256|  |   WV07|  400|
|   AWND|   9177012|  |   SX14|  730|
|   WSF2|   7348055|  |   WV01| 2950|
|   WDF2|   7347674|  |   SN34| 3023|
|   WSF5|   7278219|  |   SN82| 3074|
|   WDF5|   7270888|  |   SN83| 3074|
|   WDMV|   5768287|  |   SX83| 3075|
|   EVAP|   5704783|  |   SN72| 3075|
|   WT03|   5599740|  |   SX82| 3075|
|   WSFG|   5591129|  |   SX81| 3075|
|   WDFG|   5488593|  |   SX72| 3075|
+-------+----------+  +-------+-----+
```

Similarly, we can group by QUALITY_FLAG and count the frequency of each assurance check. The output is as shown in the table below. There are 2919648482 rows that did not fail any assurance check. By computing ((2928664265-2919648482)/2928664265)*100, we can know that about 0.3 percent of the data failed at least one kind of check or needed further investigation, which I reckon is tolerant and we can trust assumptions made based on the output of a statistical model that is trained on the dataset.

QUALITY_FLAG	null	I	D	L	S	K	O	G	X	Z
count	2919648482	6328095	1226489	329631	300290	215752	174224	137228	135629	96725|

Conclusion

This assignment has 3 sections, processing, analysis and challenges. The first section is where we explore the GHCN dataset and combine the daily climate summaries with the metadata tables, joining on station, state, and country. In analysis section, we dug deep in stations and daily data and created a user defined function that computes the geographical distance between two stations so we could have a better understanding of the efficiency when loading and applying transformations to data and generated temperature for all the country and precipitation for New Zealand, which allowed us to do some visualisations. In the last section, we checked the quality of the specific observations that each station has collected and the entire dataset by looking into daily data.

References

1. “Global Historical Climatology Network (GHCN).” National Climatic Data Center, www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets/global-historical-climatology-network-ghcn.

2.“Beginning Apache Pig.” Google Books, Google, books.google.co.nz/books?id=tC2yDQAAQBAJ&pg=PA204&lpg=PA204&dq=Since+Parquet+performs+better+compression+and+has+a+good+read+performance,+we+are+storing+the+outputs+with+parquet+format.&source=bl&ots=nIcRH7at93&sig=ACfU3U3DgdSMXLFyPfqevqKChtqovXq1Qw&hl=en&sa=X&ved=2ahUKEwj4_b7Xjf3oAhU-zDgGHYGsD4UQ6AEwAHoECA4QAQ#v=onepage&q=Since Parquet performs better compression and has a good read performance, we are storing the outputs with parquet format.&f=false.