ASSIGNMENT 4

**Task**

**(1)** Building on the simple WordCount example done in class and Hadoop tutorial, your task is to perform simple processing on provided COVID-19 dataset.

* + The first task is to count the **total number of reported cases** for every country/location till April 8th, 2020 (NOTE: There data does contain case rows for Dec 2019, you'll have to filter that data)
  + Name your program Covid19\_1.java
  + Program arguments description
    - The HDFS path to your input data file
    - [true | false] Include "World" data in the result or not. True will include total number of reported cases for "World" in the result, False will ignore the rows with location/country = world
    - The HDFS output path for your program. (NOTE: You should remove the output path after every execution of your program. Hadoop cannot start a job if output directory is already created)

**(2)** General analysis usually require processing in multiple modes. For instance, for COVID-19 data, analysis can be performed on different date ranges.

* + Your second task is to modify your program to report **total number of deaths** for every location/country in between a given range of dates.
* Name your program Covid19\_2.java
* Note that you'll have to perform error handling for invalid dates. Input dataset contains data from Dec, 2019 to April, 2020. Perform your error handling accordingly
* Also, your result should contain data including start and end dates
* Program arguments description
  + The HDFS path to your input data
  + Start date (YYYY-MM-DD)
  + End date (YYYY-MM-DD)
  + The HDFS output path for your program. (NOTE: You should remove the output path after every execution of your program. Hadoop cannot start a job if output directory is already created)

**(3)** Often times we can encounter cases where we have to join multiple datasets for analysis. One particular case is when one of the joining dataset is static and may have to be read frequently multiple times. In such cases, Hadoop provides [DistributedCache](https://hadoop.apache.org/docs/r2.6.3/api/org/apache/hadoop/filecache/DistributedCache.html) to efficiently manage read-only files and avoid unnecessary copying.

* Your third task is to output the total number of cases per 1 million population for every country
* Name your program Covid19\_3.java
* Add population.csv file to Hadoop DistributedCache (See Examples)
* Use formula

(total\_number\_of\_country\_cases/country\_population) \* 1,000,000

* Program arguments description
  + The HDFS path to your input data file (covid19\_full\_data.csv)
  + The HDFS path to populations.csv
  + The HDFS output path for your program. (NOTE: You should remove the output path after every execution of your program. Hadoop cannot start a job if output directory is already created)

**Spark Tasks:**

**(1)** Implement Task-2 on Spark using the same dataset with same arguments

**(2)**Implement Task-3 on Spark using the same dataset with same arguments

**(3)** Time and compare performance of your Spark and Hadoop implementations for Task-2 and Task-3

**Time and Performance Comparison**

Task1:Hadoop (count the number of new cases in a location)

real 0m20.738s

user 0m4.705s

sys 0m0.424s

Task2: (Count the total number of deaths in a location between an interval)

Hadoop

real 0m28.252s

user 0m5.183s

sys 0m0.566s

Spark

real 0m23.016s

user 0m14.337s

sys 0m0.932s

As we can see from the above output that sparks takes less time(real time) as compared to Hadoop.

Task2: (Count the total number of deaths in a location between an interval)

Hadoop

real 0m32.309s

user 0m4.788s

sys 0m1.396s

Spark

real 0m23.045s

user 0m14.242s

sys 0m0.943s

Similarly in this task as well, we can see that spark is faster than Hadoop .