

**HIGH PERFORMANCE COMPUTATIONAL INFRASTRUCTURES**

**CS5810**

**Individual Courswork Assessment: Hadoop MapReduce Practical Case**

**ID: 2351044**

**Academic year 2023/24**

Table Of Contents

* 1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  2. Problem Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  3. Associated Dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2.1 Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

2.2 Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3 Results, Evaluation & Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

Appendix . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

**INTRODUCTION, PROBLEM DESCRIPTION & ASSOCIATED DATASET**

INTRODUCTION

High Performance Computational Infrastructures (HPCIs) possess the needed computational power and storage capabilities to address several important computational limitations in modern data analytics. Via parallel computing techniques, HPCIs effectively distribute computational tasks amongst multiple nodes, notably improving their performance and efficiency. Additionally, they are designed to adapt to increasing computational demands, to maintain capability of handling more complex tasks. Utilizing distributed storage systems which enable storage and management of large datasets across various storage nodes, they are also able to accelerate scientific simulations, machine learning algorithms and data analytics processes by balancing the distribution of the workload. Redundancy, checkpointing and recovery strategies are some of the mechanisms used to manage their scale and complexity. These mechanisms facilitate system reliability and availability, mitigating the impact of hardware failures or system errors.

HPCIs allow researchers, engineers and data analysts to pursue novel innovations and discoveries in science, engineering, finance, and technology. Within science, HPCIs contribute significantly to simulations, modelling and data analysis. They are also instrumental in engineering and design workflows as well as financial risk analysis.

PROBLEM DESCRIPTION

The research question aimed to analyse air quality data across different countries and evaluate the predominant pollutant for each country. By grouping countries based on the predominant pollutant, the study could provide insights into regional air pollution patterns and the factors influencing pollutant concentrations. To simplify, the research question asked, “what are the predominant air pollutants across each country?" The results could inform policymakers and environmental scientists in devising interventions that could address specific air quality challenges within different regions.

ASSOCIATED DATASET

The global\_air\_pollution\_data dataset derived from Kaggle, with URL: https://www.kaggle.com/datasets/hasibalmuzdadid/global-air-pollution-dataset, contains 23,036 rows after elimination of missing instances, 175 countries and over 300 cities, delivering a cohesive view of global air quality dynamics. The pollutants considered within the dataset include carbon monoxide (CO), ozone (O3), nitrogen dioxide (NO2), and particulate matter (PM2.5). The dataset includes data from the current year and is seen as a key asset to policymakers and environmental scientists. Insights derived could potentially contribute to promoting a more sustainable environment. The metadata is as follows:

country\_name: Name of the Country

city\_name: Name of the City

aqi\_value: Overall Air Quality Index (AQI) value of the city

aqi\_category: Overall AQI category of the city

co\_aqi\_value: AQI value of Carbon Monoxide of the city

co\_aqi\_category: AQI category of Carbon Monoxide of the city

ozone\_aqi\_value: AQI value of Ozone of the city

ozone\_aqi\_category: AQI category of Ozone of the city

no2\_aqi\_value: AQI value of Nitrogen Dioxide of the city

no2\_aqi\_category: AQI category of Nitrogen Dioxide of the city

pm2.5\_aqi\_value: AQI value of Particulate Matter with a diameter of 2.5 micrometres or less of the city

pm2.5\_aqi\_category: AQI category of Particulate Matter with a diameter of 2.5 micrometres or less of the city

Finally, it is important to note that 427 instances from this dataset were removed in RStudio due to the country\_name being missing, and the dataset was renamed to Global\_air\_pollution\_data\_cleaned.

**DESIGN & IMPLEMENTATION**

DESIGN

Hadoop distributed file system (HDFS) was utilized in this computational task, along with its MapReduce framework. MapReduce broke down the computational task into a mapper phase which extracted the relevant columns to produce key-value pairs which were then fed to the reducer phase, which calculated the highest pollutant proportion of AQI per country and outputs a list of all countries with the most culpable pollutant accompanying it. It was also important to note that the AQI values were given in relation to cities within each country, hence the mapper was key in collating the countries for each records. The task was implemented on a Hadoop cluster in Google Colab and was saved as a Jupyter source file. The mapper and reducer files were saved as mapper-CS5810\_2351044.py and reducer-CS5810\_2351044.py respectively. A combiner was not needed for this task.

IMPLEMENTATION

Mapper:

The column titles are ignored.

    if not header\_skipped:

        header\_skipped = True

        continue

Splitting the columns of the csv file using commas

    columns = line.strip().split(',')

Assigning the relevant columns to an associated object

    country = columns[0]

    co\_aqi\_value = columns[4]

    ozone\_aqi\_value = columns[6]

    no2\_aqi\_value = columns[8]

    pm25\_aqi\_value = columns[10]

Defining the key-value pairs with the column objects with country\_name being separated from each pollutant and its associated AQI value with the “\t” separator.

    print(f"{country}\tCO,{co\_aqi\_value}")

    print(f"{country}\tO3,{ozone\_aqi\_value}")

    print(f"{country}\tNO2,{no2\_aqi\_value}")

    print(f"{country}\tPM2.5,{pm25\_aqi\_value}")

Reducer:

Initializing the variable to keep track of countries being processed, a dictionary for each pollutant to add up their AQI value. Then initializing a variable to count the number of records.

current\_country = None

total\_aqi = {'CO': 0, 'O3': 0, 'NO2': 0, 'PM2.5': 0}

count = 0

Splitting the mapper output by the “\t” separator, then splitting the pollutant and AQI value by “,”.

    country, pollutant\_aqi = line.strip().split('\t')

    pollutant, aqi\_value = pollutant\_aqi.split(',')

The AQI value is converted to a floating-point number

    aqi\_value = float(aqi\_value)

The current\_country then begins by taking the country object created in the mapper, selecting each country from the dataset

    if current\_country is None:

        current\_country = country

Calculating average AQI value for each pollutant when the country changes by dividing the current total AQI value for each pollutant by the number of recorded instances.

    if current\_country != country:

        avg\_aqi = {pollutant: total\_aqi[pollutant] / count for pollutant in total\_aqi}

Determining the predominant pollutant for the current country based on the pollutant who’s average AQI is the highest.

        predominant\_pollutant = max(avg\_aqi, key=avg\_aqi.get)

Printing the country name and its predominant pollutant.

       print(f"{current\_country} {predominant\_pollutant}")

Resetting the variables to process the next country.

   current\_country = country

        total\_aqi = {'CO': 0, 'O3': 0, 'NO2': 0, 'PM2.5': 0}

        count = 0

Accumulating the AQI value for the current pollutant and increasing the count records for the current country.

   total\_aqi[pollutant] += aqi\_value

    count += 1

Calculating the average AQI per pollutant for the final country.

if current\_country is not None:

    avg\_aqi = {pollutant: total\_aqi[pollutant] / count for pollutant in total\_aqi}

Determining the predominant pollutant for the final country.

    predominant\_pollutant = max(avg\_aqi, key=avg\_aqi.get)

Printing reducer output.

    print(f"{current\_country} {predominant\_pollutant}")

**RESULTS, EVALUATION & DISCUSSION**

The results suggested that most countries needed to worry about Particulate Matter with a diameter of 2.5 micrometres (PM2.5) while a few had to worry about ozone (O3) more than any other pollutant. This information would be vital to most environmental researchers and policymakers. It was also important to note the effect of eliminating several rows from the dataset due to missing values on the final result. HPCIs are essential components in modern organizational environments. They store, generate, collect and analyse vast amounts of data, and in this computation, a vast amount of data was collected from the global air pollution dataset, analysed using MapReduce and a large amount of meaningful output was generated by the reducer to derive insights which could inform national policy on addressing air pollution worldwide. The Hadoop framework leveraged clusters distributed the data processing tasks across multiple nodes for parallel processing, to aid in efficient computation and handling of the large amounts of data. HPCIs empower organizations to use the power of big data for insights, innovation, and competitive advantage by understanding the underlying concepts, theories, and appropriate use cases to guide in the design and implementation of scalable, reliable and efficient data processing to meet their needs.

**APPENDIX**

The appendix material was submitted as a zipped/archived folder containing the following files:

1. Global\_air\_pollution\_data\_subset.csv – Subset of dataset used with 500 rows

2. Jup\_CS5810\_2351044.ipynb – Jupyter source file ran in google colab

3. mapper-CS5810\_2351044.py – the mapper python code

4. reducer-CS5810\_2351044.py – the reducer python code

5. CS5810\_2351044\_result – output generated by MapReduce into the HDFS