CMPE 275: Mini Project 2 Report
AQI Average Progression in Time across US sites

Team Members:
Rajath Nattoji Rajaram (017426097)
Ruthvik Kamarasu (017422847)
Naga Varun Bathina (017434261)
Rohit Uday Vardam (017437433)

Introduction

We designed a program to process and analyze Air Quality Index (AQI) data across multiple locations over time, using advanced programming techniques and technologies. The codebase integrates several key programming concepts and libraries, such as Multi-Processing Interface (MPI) for distributed computing, OpenMP for parallelism, shared memory for inter-process communication, and filesystem operations for data management. At its core, the program aims to efficiently parse, process, and aggregate large datasets of air quality measurements, eventually producing consolidated reports on the AQI for various locations.

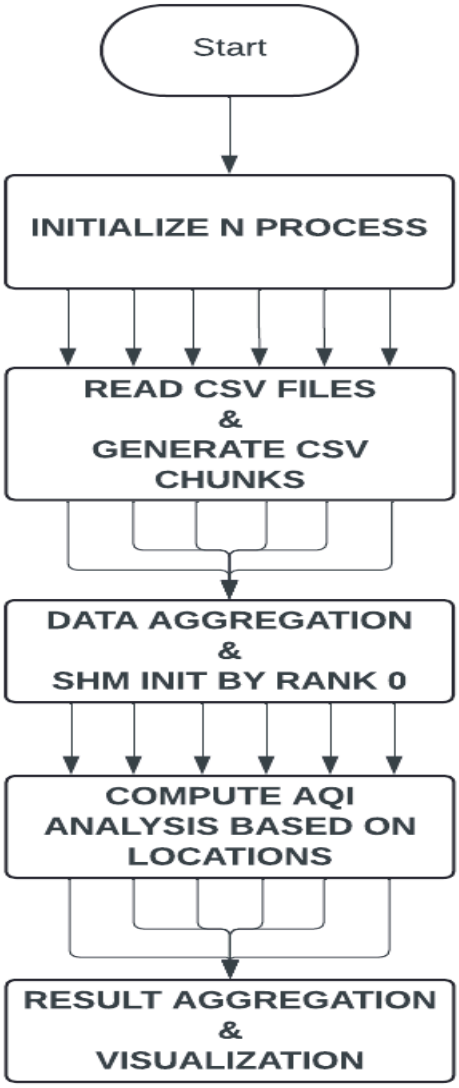
Approach

We took a divided approach. The tasks were divided into several distinct phases: data parsing, parallel processing, data aggregation, and report generation. Initially, the program reads and parses AQI data from CSV files, which contain air quality measurements across different cities and dates. This parsing involves extracting relevant information and structuring it into a usable format for further processing.

To handle the computational demands of processing large volumes of data, we employed MPI to distribute tasks across multiple processes. Each process is responsible for a subset of the data, ensuring that the workload is evenly distributed and processed in parallel. Further parallelism is achieved within each process using OpenMP, which allows for concurrent execution of tasks such as file reading and data aggregation across multiple threads.

Shared memory is utilized to aggregate the processed data from all parallel processes. This method ensures efficient communication and data sharing among processes, leading to the final phase of report generation. In this phase, the aggregated data is formatted into a comprehensive report, showcasing the average AQI values for each location over time.

Methodology



Flow Diagram

- **Data Parsing and Structuring**: Leveraging C++'s STL for efficient data manipulation and storage, specifically using std::vector, std::string, and custom structs to model the AQI data.
- **Multi-processing with MPI**: Implementing MPI to divide the workload among multiple processes. This includes the initiation of MPI, data distribution, and collective operations for data gathering.

- **Parallel Processing with OpenMP**: Utilizing OpenMP directives to parallelize the processing of data within each MPI process, significantly speeding up tasks such as file reading and data parsing.
- **Shared Memory for Data Aggregation**: Using System V shared memory APIs to share the final aggregated data among all processes, enabling the consolidation of results for final output generation.
- **Filesystem Operations for Data Management**: Employing the std::filesystem library to navigate through directories and files, facilitating the program's ability to dynamically access and process datasets.
- **Performance Measurement**: Incorporating std::chrono for high-resolution timing, allowing for the measurement of the program's execution time and the evaluation of its performance efficiency.

The integration of these techniques and technologies showcases a comprehensive approach to solving the problem of large-scale AQI data analysis, emphasizing parallelism, efficiency, and scalability.

Features

Location-based segregation: Given CSV files are divided on the basis of geographic location for example you could say, the location of San Jose would be spread across several CSV files. Initially, the segregation is performed by taking into consideration the location where the AQI is measured. The average AQI value of the particular location is what we're aiming to achieve. So, in order to do that, we need to narrow down all the files to new CSV files for each location

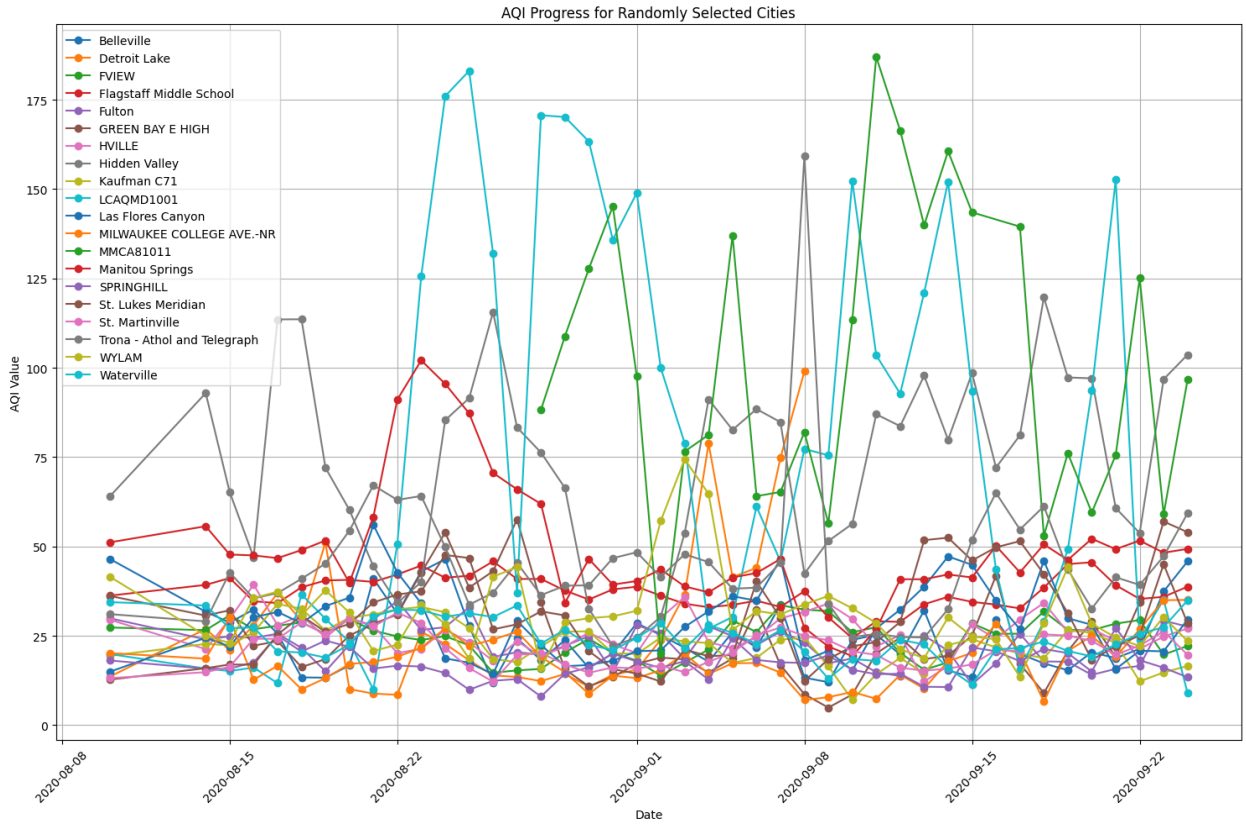
Visualizations using Pandas: Initially, to make sense of data, Pandas Dataframes are utilized to plot the AQI index value of each geographical location over the time for each date respectively. This could be further analyzed to narrow down trends which could help in handling the Air Quality and identifying the major causing pollutants and eventually handling them.

Multithreading to read files: Unlike sequentially handling data which could ultimately take a lot of time and resources and is not scalable, MPI helps in handling this data using multiple processes simultaneously. Each process takes over a certain number of files assigned and the process handles each file by creating multiple threads that increase efficiency in handling this process as the trend in the graphs evidently portrays. Each thread is responsible for a certain number of rows of a particular file and segregates them into a particular required location that could be averaged down later.

Leader-based Aggregation: All the segregated data based on geographic location need to be aggregated with the help of MPI. Basically, the multiple threads which are 4 in our case that processed data need to be aggregated through another aggregating process that gets a unique row for each location with date and AQI Average value as the other 2 columns. This is stored in the form of raw string data in Rank 0, which is the leader that is responsible for handling all processes and aggregating. Then further, the final CSV file is generated through this raw string data consisting of all unique 1397 locations with AQI average value for a particular date.

Visualizations

Average AQI progression for 20 randomly selected cities



Performance Sequential vs Hybrid with multi-threading and parallel processing