# **High Performance Computing Simulation of Air Now Data - California 2015 - 2024**

This report presents a comprehensive analysis of a project aimed at simulating air quality enforcement scenarios using High Performance Computing (HPC) techniques. The project utilizes a combination of C/C++ and Python programming languages, leveraging parallel computing technologies such as OpenMP and MPI to model scenarios over various time scales—ranging from a day to a year. Through the assessment of the project's codebase, this report identifies the methodologies employed, evaluates the use of HPC techniques, and examines the scalability, data handling, and visualization strategies. Recommendations for enhancing the project's performance, scalability, and data visualization capabilities are provided.

## **Introduction**

Air quality management and enforcement are critical for environmental protection and public health. Simulating these scenarios can provide valuable insights into the effectiveness of various policies and interventions. This project seeks to model air quality enforcement scenarios, focusing on particulate matter (PM10, PM2.5, NO2, Ozon, CO, SO2, Pb) concentrations, using simulations that span from a single day to a year. The simulations are executed in a mock High Performance Computing environment, utilizing C/C++ for computational tasks and Python for initialization and visualization.

### **Objectives**

* To simulate air quality scenarios with varying time scales.
* To employ HPC techniques for parallel data processing and simulation.
* To visualize the simulation outcomes for analysis and reporting.

## **Methodology**

The project's methodology encompasses parallel programming for simulation and data processing, data handling for simulations, and visualization of outcomes. C/C++ files are used for the core simulation tasks, leveraging MPI for process communication and OpenMP for threading, while Python scripts handle data preprocessing and generate heatmaps for visual analysis.

### **Parallel Programming Techniques**

The simulations employ MPI and OpenMP to achieve parallelism across distributed systems and within single nodes, respectively. This hybrid approach allows the project to utilize computing resources efficiently, handling large datasets and complex calculations necessary for the simulations.

#### C++ Simulations

* MPI: Enables distributed computing across multiple nodes, facilitating large-scale simulations.
* OpenMP: Utilized within nodes to parallelize computations across multiple cores, enhancing performance.
* Shared Memory: Used Pragma to parallelize loops.

#### Python Visualization

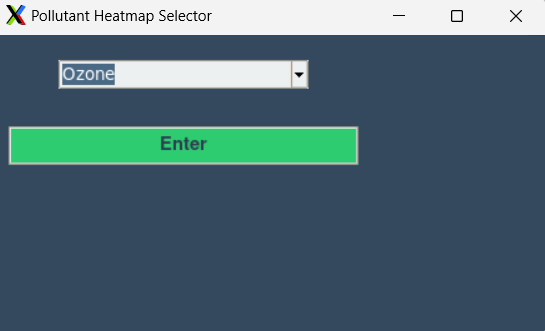
* mpi4py: Employs MPI in Python, allowing for parallel processing in data visualization tasks, specifically in generating heatmaps of pollutant concentrations.

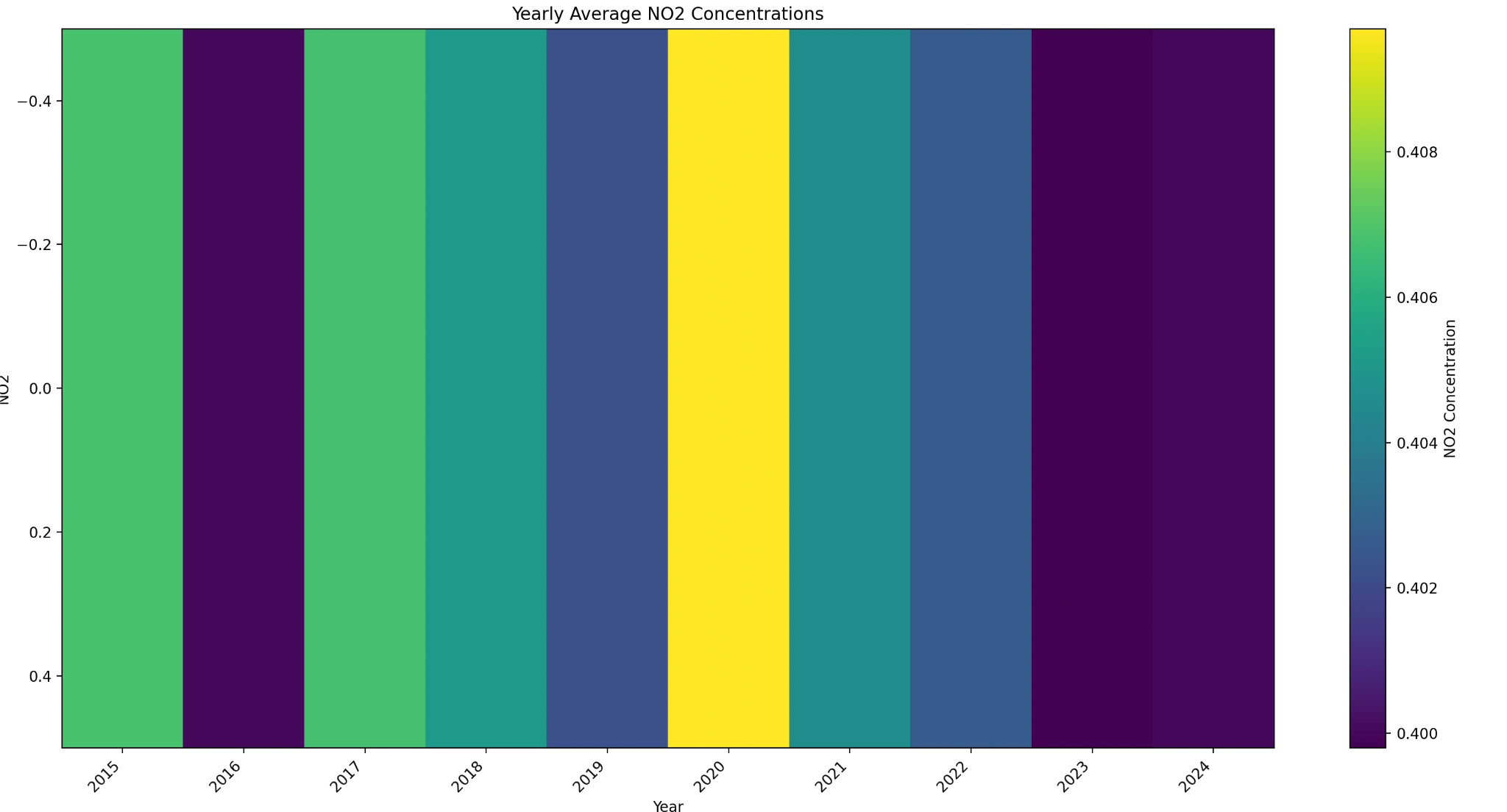
### **Data Handling and Simulation Logic**

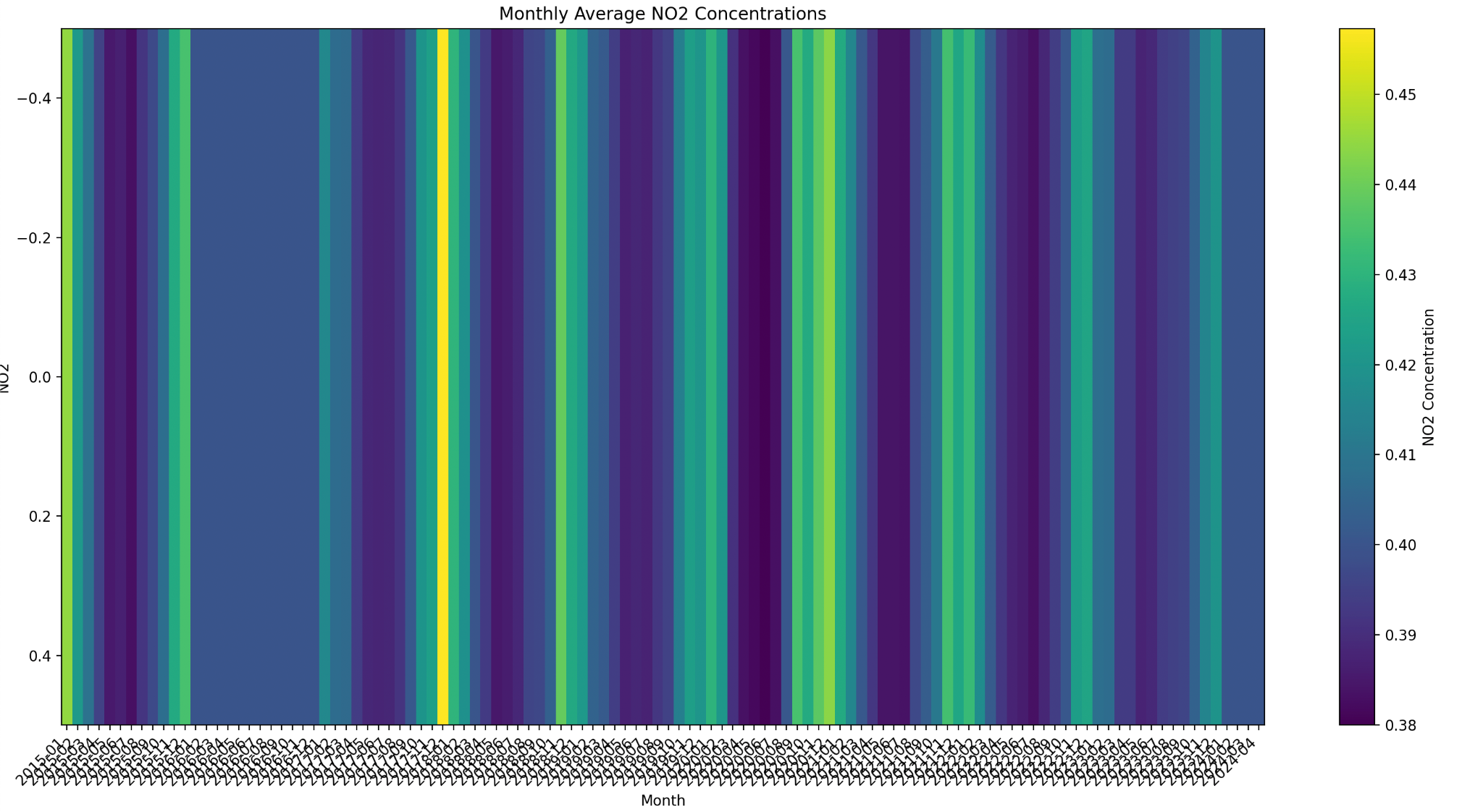
The project implements sophisticated data handling techniques to manage input data for simulations and visualizations. Simulations calculate global averages of PM concentrations, requiring accurate and efficient data processing methods to ensure realistic and meaningful outcomes.

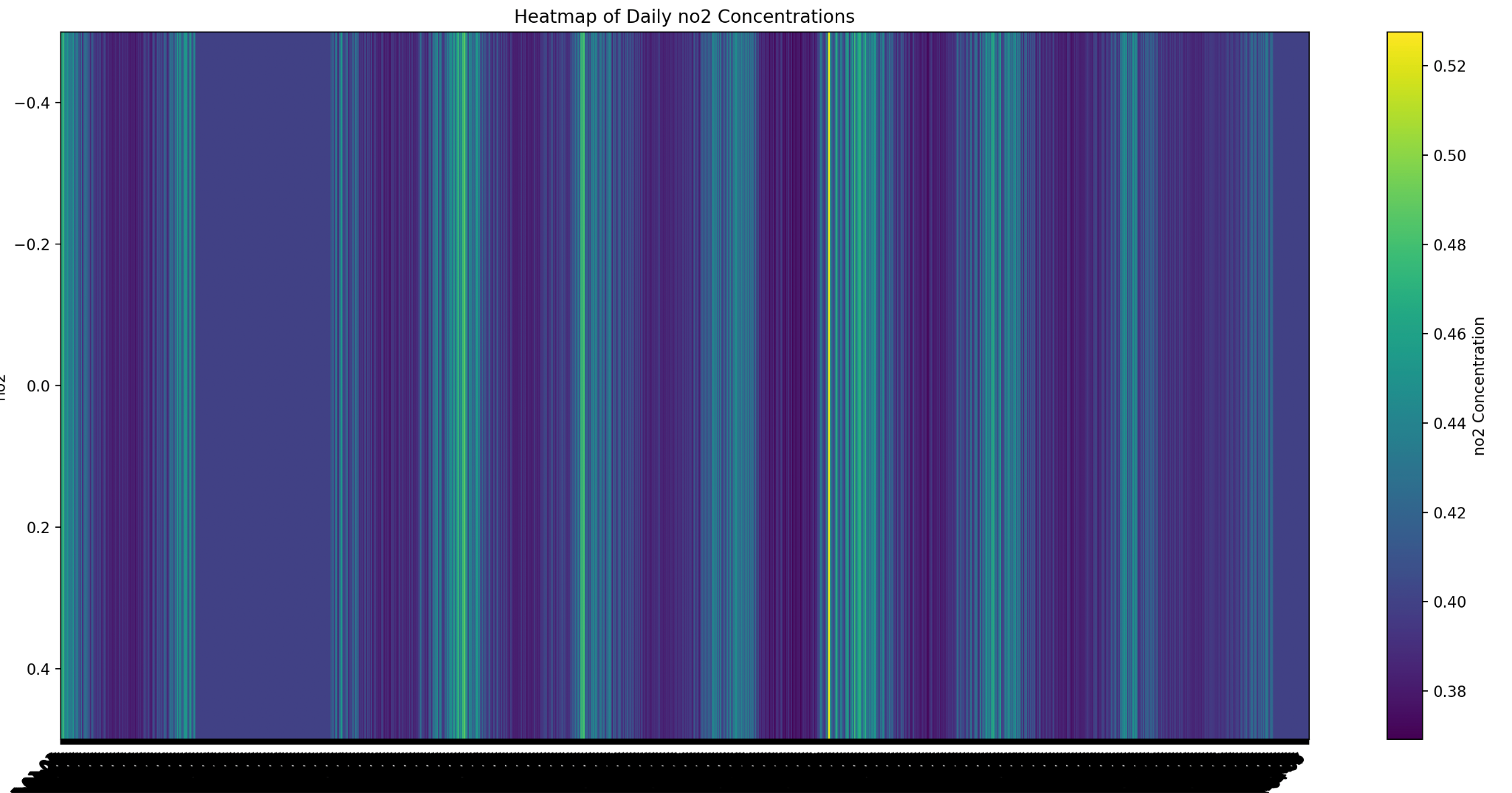
### **Visualization and Output**

Python scripts generate heatmaps using matplotlib, presenting an intuitive visual representation of PM concentrations across different scenarios. This visualization facilitates easy interpretation of simulation results, highlighting areas of concern and effectiveness of air quality enforcement strategies.









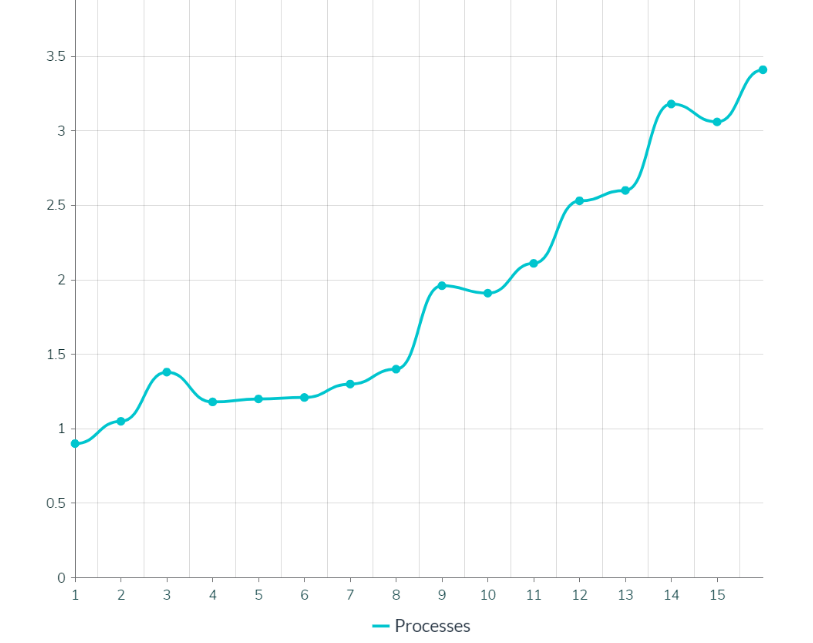
## 

## 

## **Analysis**

### **Scalability and Performance**

The project's design and implementation suggest a strong foundation for scalability, essential for HPC applications. However, performance could be hindered by bottlenecks such as data I/O operations and communication overhead. Identifying and addressing these bottlenecks is crucial for maximizing efficiency.



| **No. of Processes** | **Time** |
| --- | --- |
| 1 | 0.9s |
| 2 | 1.05s |
| 3 | 1.38s |
| 4 | 1.18s |
| 5 | 1.20s |
| 6 | 1.21s |
| 7 | 1.3s |
| 8 | 1.4s |
| 9 | 1.96s |
| 10 | 1.91s |
| 11 | 2.11s |
| 12 | 2.53s |
| 13 | 2.6s |
| 14 | 3.18s |
| 15 | 3.06s |
| 16 | 3.41s |

| **No. of Processes** | **Time** |
| --- | --- |
| 8 | 2.40s |
| 50 | 14.16s |
| 100 | 30.3s |
| 150 | 47.8s |
| 200 | 65.23s |
| 250 | 81.8s |
| 300 | 102.99s |

### **Best Practices and Design**

The project adheres to several best practices in parallel programming and HPC application development. Nevertheless, further improvements in code structure, such as increased modularity and encapsulation, could enhance maintainability and extensibility.

## **Future Enhancements**

* Data Handling Optimization: Implement advanced data partitioning and load balancing strategies to improve scalability and performance.
* Visualization Enhancements: Explore more sophisticated visualization techniques to provide deeper insights and more intuitive analysis of the simulation data.
* Code Refactoring: Increase the codebase's modularity and encapsulation to improve code maintainability and facilitate future enhancements.

### **Compiling and Running C++ Files**

1. Compilation:

* To compile a C++ file, such as parameter\_simulation.cpp, which represents a simulation component of the project, use the mpic++ compiler. This compiler is part of the MPI (Message Passing Interface) toolkit, enabling the code to utilize distributed computing capabilities.
* **mpic++ -o <simulation\_filename> <simulation\_filename.cpp>**
* This command compiles the parameter\_simulation.cpp file and generates an executable named parameter\_simulation.

1. Running the Simulation:

* To run the compiled simulation, the mpiexec command is used, specifying the number of processes to distribute the task across. For example, to run the simulation using 4 processes:
* **mpiexec -n 4 ./<simulation\_file>**

1. Scaling the Simulation:

* If your system supports it, you can scale the simulation to use more processes than the number of cores available. This can be useful for testing scalability and performance in a constrained environment. The --oversubscribe option allows you to run more processes than available cores:
* **mpiexec --oversubscribe -n 250 ./<simulation\_executable\_file>**
* This command attempts to run the simulation with max 250 processes, which can be useful for testing the simulation's scalability and performance under high load.

### **Running Python Scripts**

1. Initialization:

* Before running simulations, it may be necessary to initialize datasets or perform pre-processing. If such tasks are handled by a Python script, such as init.py, you can execute it directly with Python:
* **python3 src/init.py**
* Ensure that the script init.py is located in the src directory. This script typically handles data loading, cleaning, and preparation tasks.

1. Running Python mpi4py Files:

* For Python scripts that utilize mpi4py for parallel processing, such as parameter\_heatmap.py for generating heatmaps of simulation results, use the mpiexec command with the -np (number of processes) option:
* **mpiexec -np 4 python3 src/<heatemap filename>**
* This command runs the parameter\_heatmap.py script in parallel using 4 processes, facilitating efficient data processing and visualization. Make sure the script is located in the src directory.

**Contributors**

Jahnavi Salammagari

[Krushika Gujarati](mailto:krushika.gujarati@sjsu.edu)

[Shreya Reddy Edulakanti](mailto:shreyareddy.edulakanti@sjsu.edu)

Faiz Ahamed Shaik

**Citation**

<https://mpitutorial.com/tutorials/>

<https://mpi4py.readthedocs.io/en/stable/>

## **Conclusion**

The project represents a significant step towards using HPC techniques to simulate and analyze air quality enforcement scenarios. By leveraging parallel computing, it achieves scalable simulations across different time scales, providing valuable insights into particulate matter concentration management. With recommended optimizations and enhancements, the project has the potential to offer even deeper insights and more efficient performance, contributing to more effective air quality management strategies.

This report has laid out the current state of the project, analyzed its strengths and areas for improvement, and provided recommendations for future development. As environmental challenges continue to evolve, such HPC applications will be invaluable in crafting strategies that ensure cleaner air and healthier communities.