Name	ID
Alaa Asharf lotfy	221001804
Nour ali elsaman	221001892
Esraa Mahmoud	221001883
Mohamed Abdelkader	221001955

1.Introduction

In recent years, High-Performance Computing (HPC) and big data processing have become essential tools driving progress in many scientific and industrial fields, such as genome analysis, artificial intelligence, physical simulations, and machine learning. Today's research challenges require enormous computing power and storage capacity, which makes it crucial to learn how to design and build distributed computing systems that can efficiently handle large and complex datasets.As part of our HPC course, we were assigned a practical project to create a Mini-HPC Cluster and a hybrid environment that combines HPC with big data processing using Apache Spark. Over four weeks, we set up a virtual environment with three nodes, ran distributed machine learning tasks using the MPI library, and analyzed real biological data with Spark tools running inside a Docker Swarm cluster.This project was a valuable learning experience that brought together theory and practice in distributed computing. It helped us improve our skills in managing complex infrastructures and using open-source tools to solve real-world problems in bioinformatics and machine learning.

The goal of this project is to train students on how to set up a distributed computing system using three virtual machines-one master and two workers-and to run distributed machine learning tasks using MPI and Python. The project also involves using Docker to create a Spark cluster for analyzing real biological data. It aims to bridge the gap between theory and practical application, helping students understand modern techniques used for big data analysis and efficient model execution .

2. Methodology

This project was divided into two main phases: the creation of a traditional Mini-HPC cluster and the deployment of a hybrid HPC-Big Data environment using Apache Spark. Each phase was carefully implemented using open-source tools and executed in a virtualized environment for flexibility and control.

3.Virtual Cluster Setup

Three Ubuntu Server 20.04 virtual machines were created using Oracle VirtualBox, designated as one master node and two worker nodes. Static IP addresses were assigned to each VM using a host-only adapter to ensure stable communication. Network connectivity was tested and verified using standard ping commands.

4 SSH Configuration and Environment Preparation

To facilitate secure and automated communication between nodes, passwordless SSH was configured from the master node to each worker node using ssh-keygen and ssh-copy-id. Subsequently, essential packages including Python, OpenMPI, and relevant Python libraries (mpi4py, scikit-learn, numpy, pandas) were installed across all nodes to support parallel machine learning tasks.

5 Distributed Machine Learning with MPI

An initial machine learning script was developed using the mpi4py library to perform parallel training on the digits dataset from scikit-learn. The data was broadcast from the master node to all worker nodes. Each node trained a Random Forest classifier on its data partition, and accuracy scores were collected and aggregated on the master node. A second MPI-based script was implemented to process a real-world leukemia gene expression dataset in the same distributed manner.

6 Hybrid HPC + Spark Cluster Deployment

In the second phase, Docker was installed on all nodes, and a Docker Swarm cluster was initialized. The Spark cluster was deployed using a Docker Compose file specifying one Spark master and two Spark worker containers. The setup was monitored and validated through the Spark Web UI.

7 Bioinformatics Analysis using PySpark

To analyze the leukemia dataset, a PySpark script was executed within the Spark cluster. The data was preprocessed using VectorAssembler to format features appropriately. A Random Forest model was trained and evaluated within the distributed environment, and the accuracy was calculated on the test set. This allowed for comparison between the MPI and Spark implementations in terms of both accuracy and execution time.

8. Results

1. Cluster Setup and Configuration

The distributed cluster was successfully set up using three virtual machines: one master and two worker nodes. Network connectivity between all nodes was verified through ping tests, confirming seamless communication across the cluster.

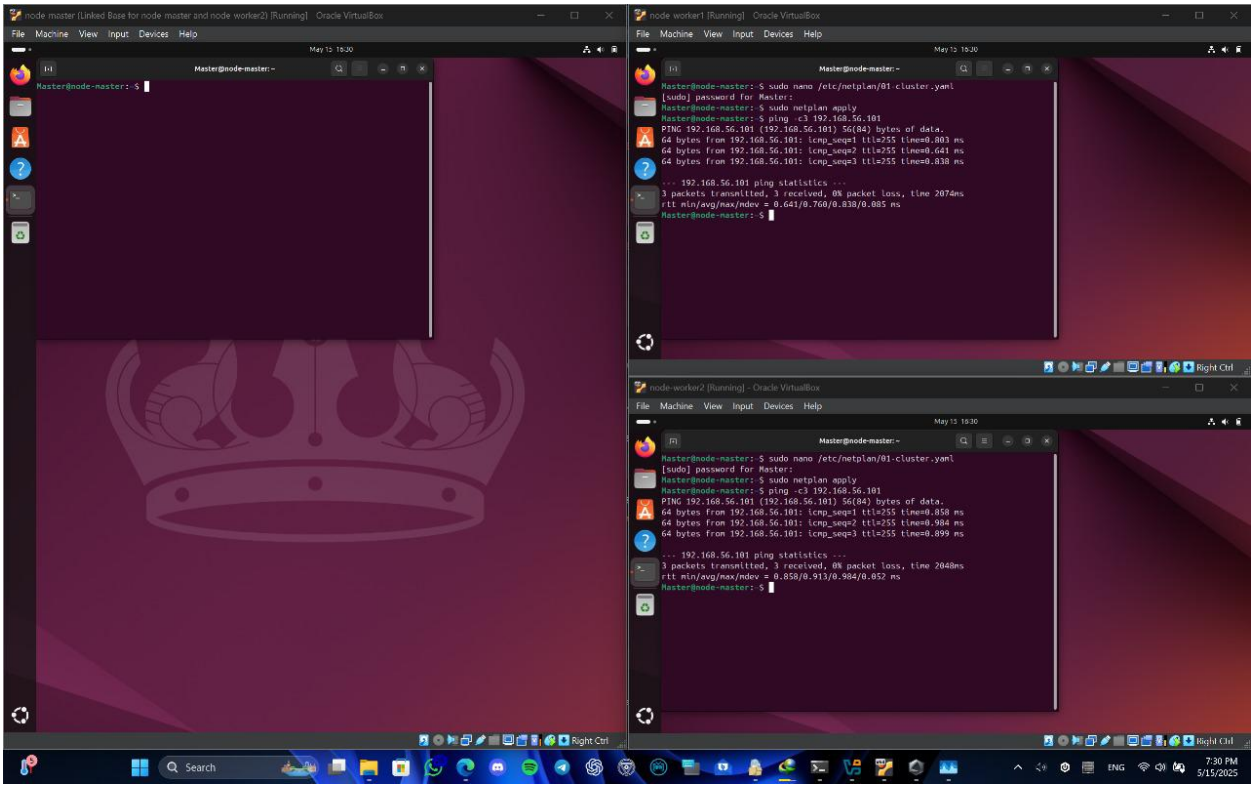
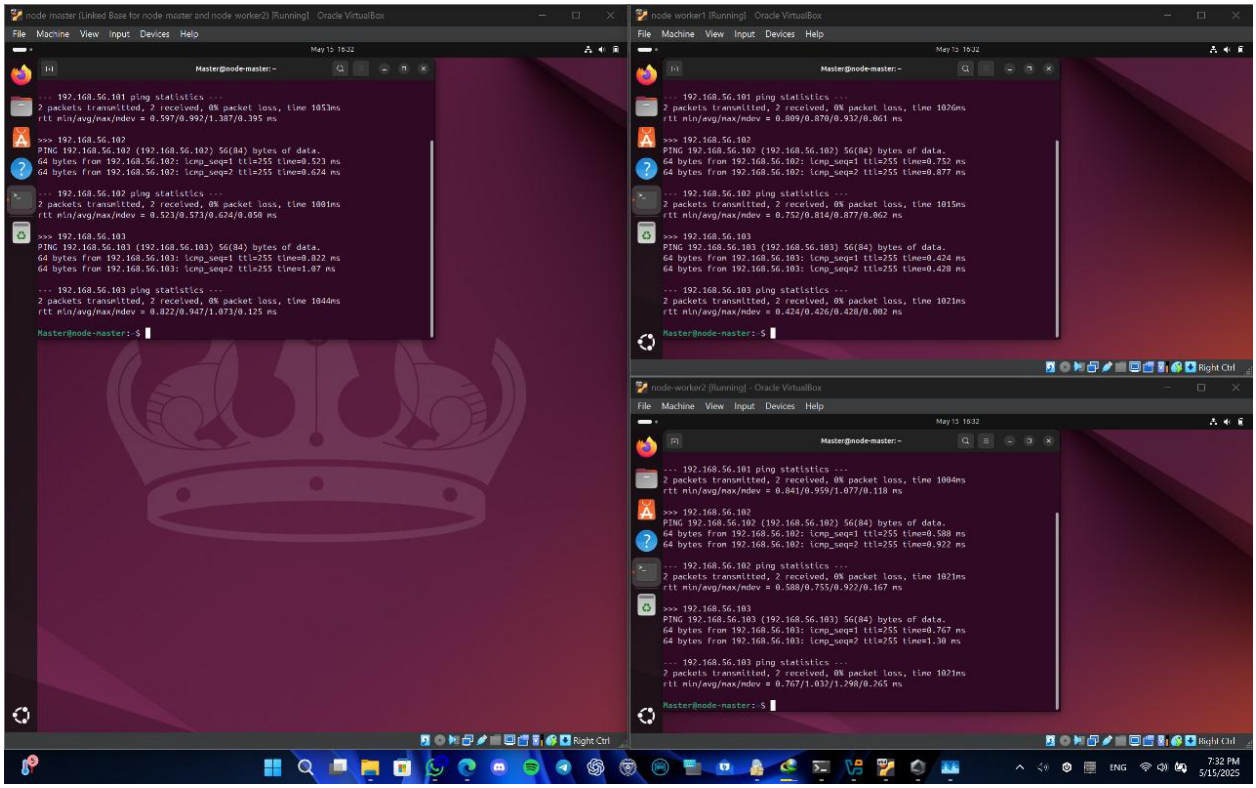


Figure 1:communication between the master node and the first worker node.



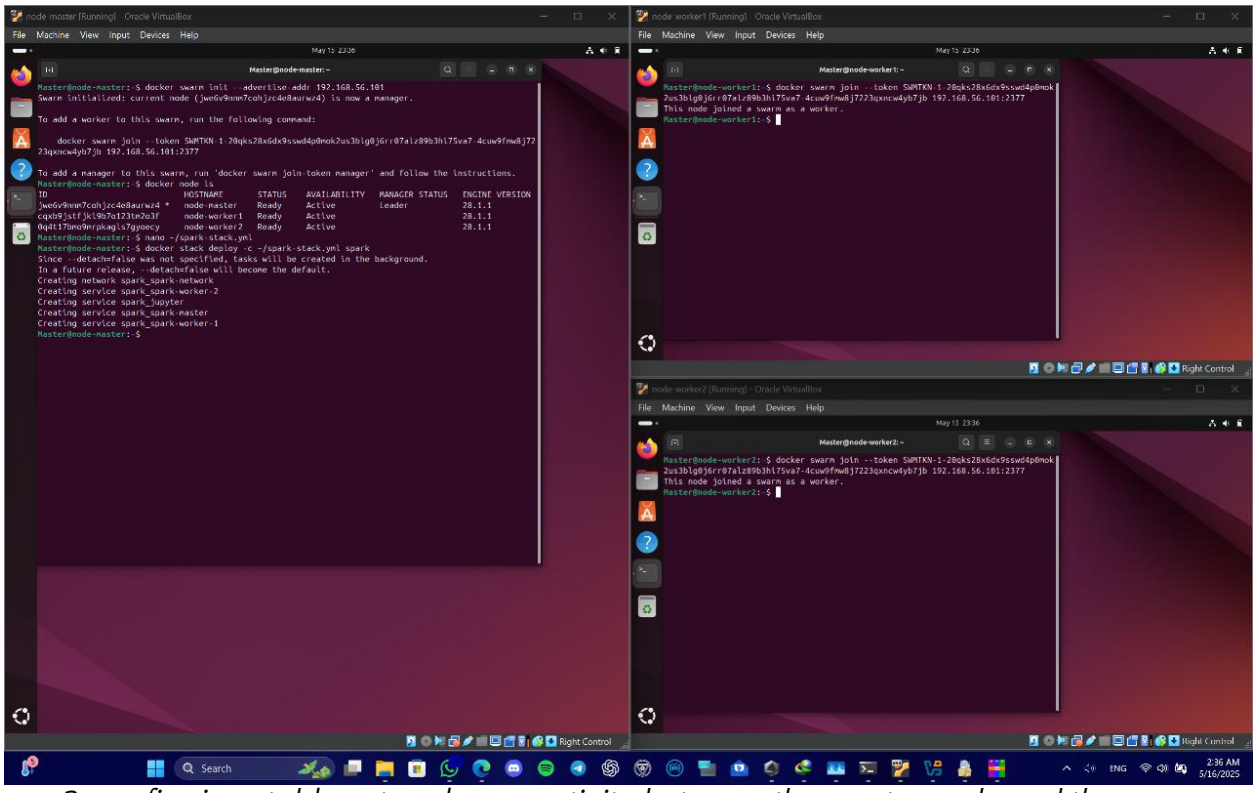


Figure 2: confirming stable network connectivity between the master node and the second worker node.

2. MPI-based Distributed Machine Learning

OpenMPI and the mpi4py library were installed on all nodes to enable distributed machine learning. Training was performed on a gene expression dataset, partitioned across the nodes. The results below display the accuracy and training time for each process, as well as the combined ensemble model performance.

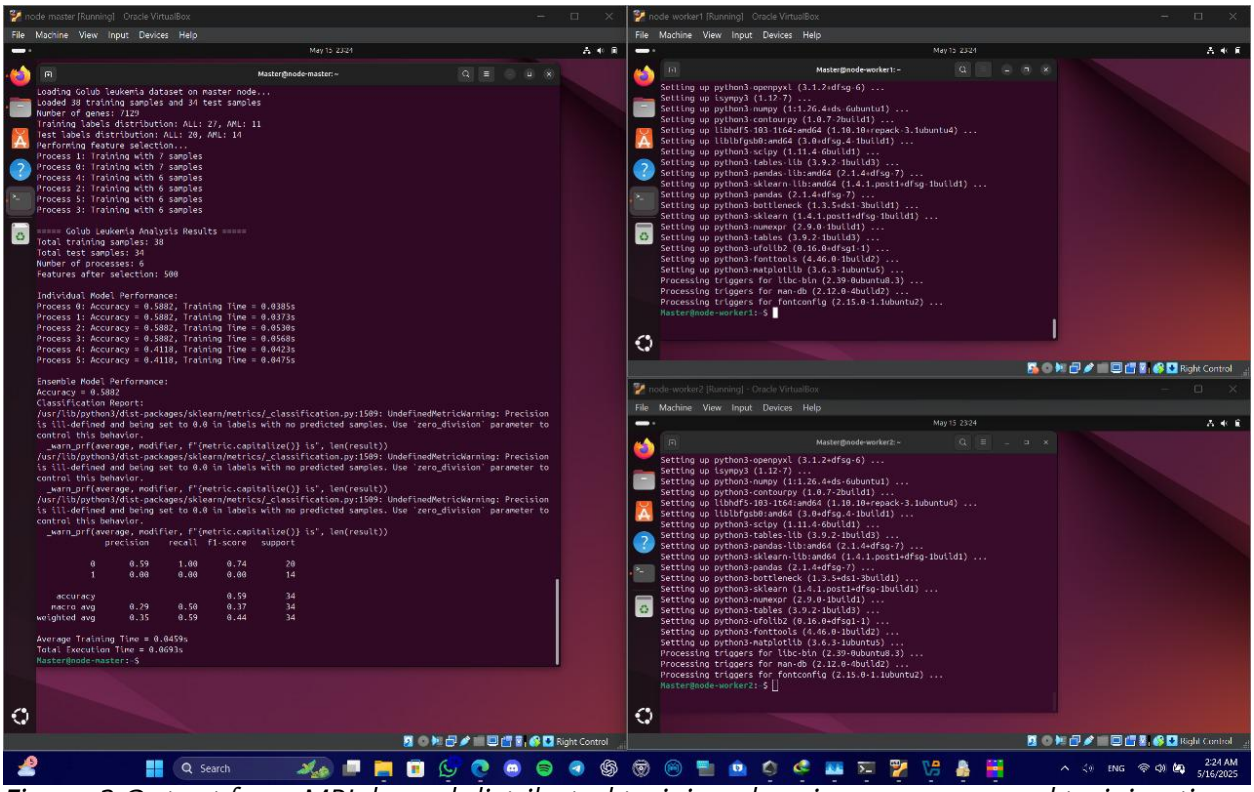


Figure 3:Output from MPI-based distributed training showing accuracy and training time for each process and the ensemble model.

3. Docker Swarm and Spark Cluster Deployment

Docker was installed across all nodes, and a Docker Swarm cluster was initialized to manage the three machines collectively. Apache Spark was deployed on this cluster using Docker Compose with a custom configuration file. The Spark master and worker nodes were confirmed to be running and connected.

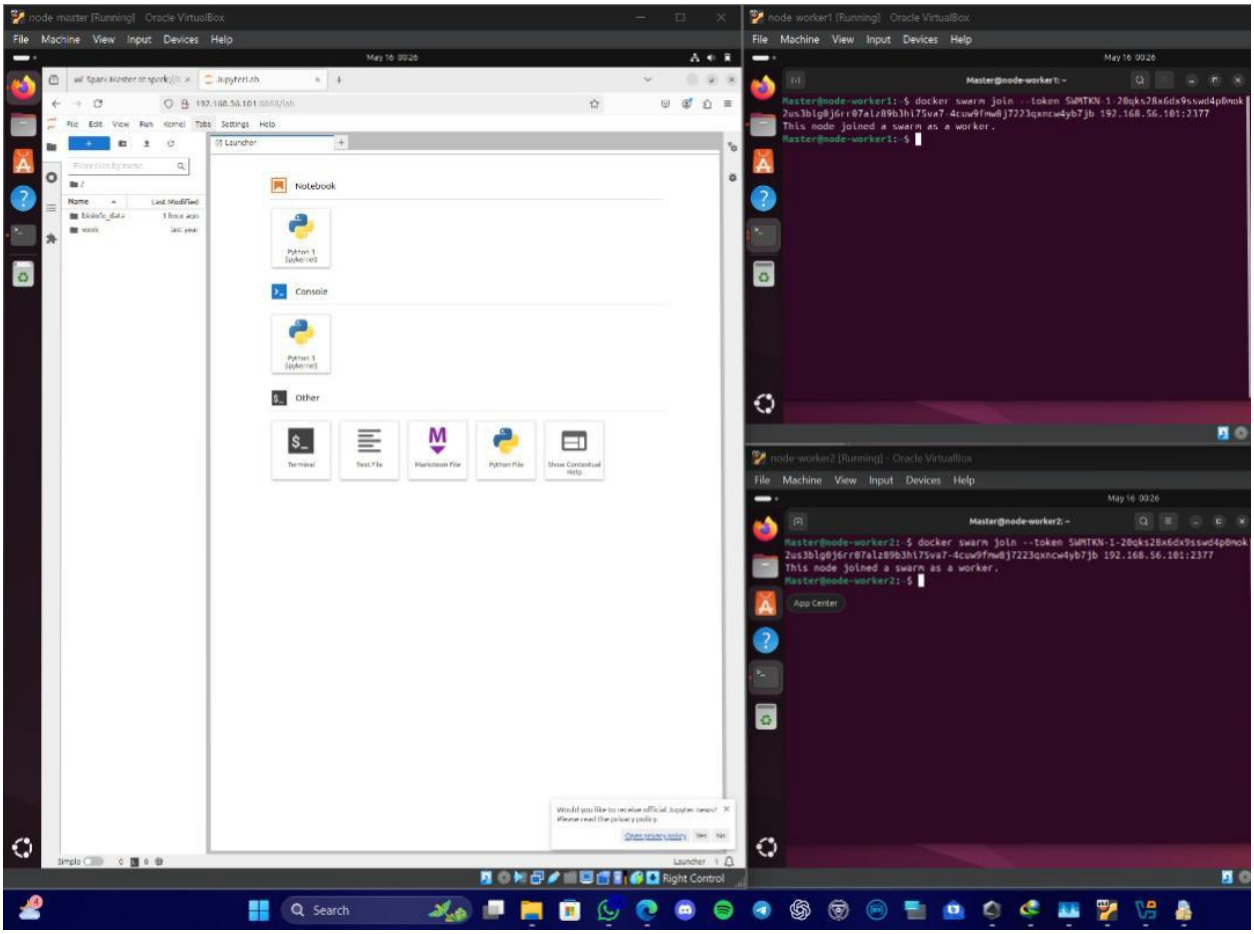


Figure 4 :JupyterLab interface running on the Spark master node, providing an environment for distributed data processing.

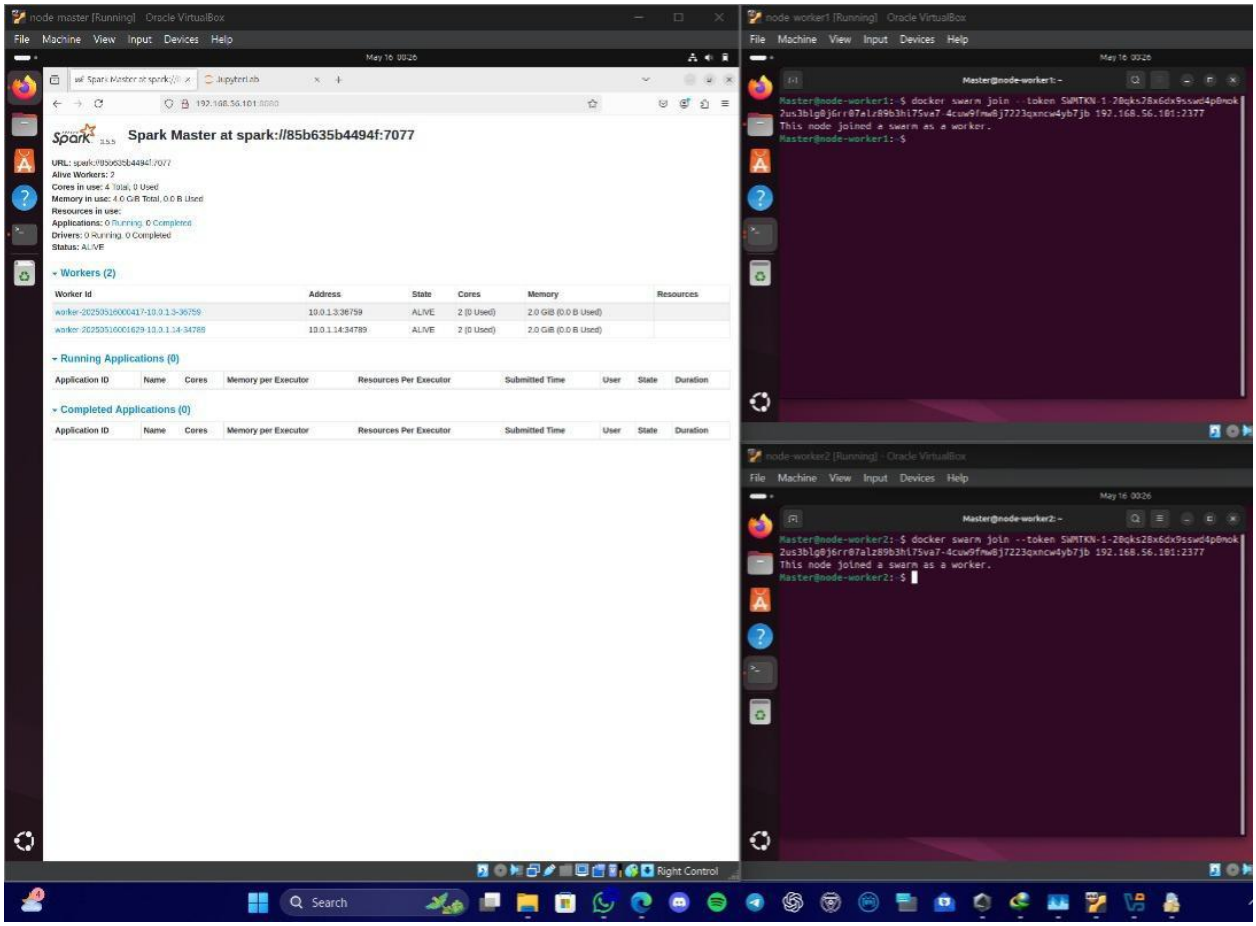
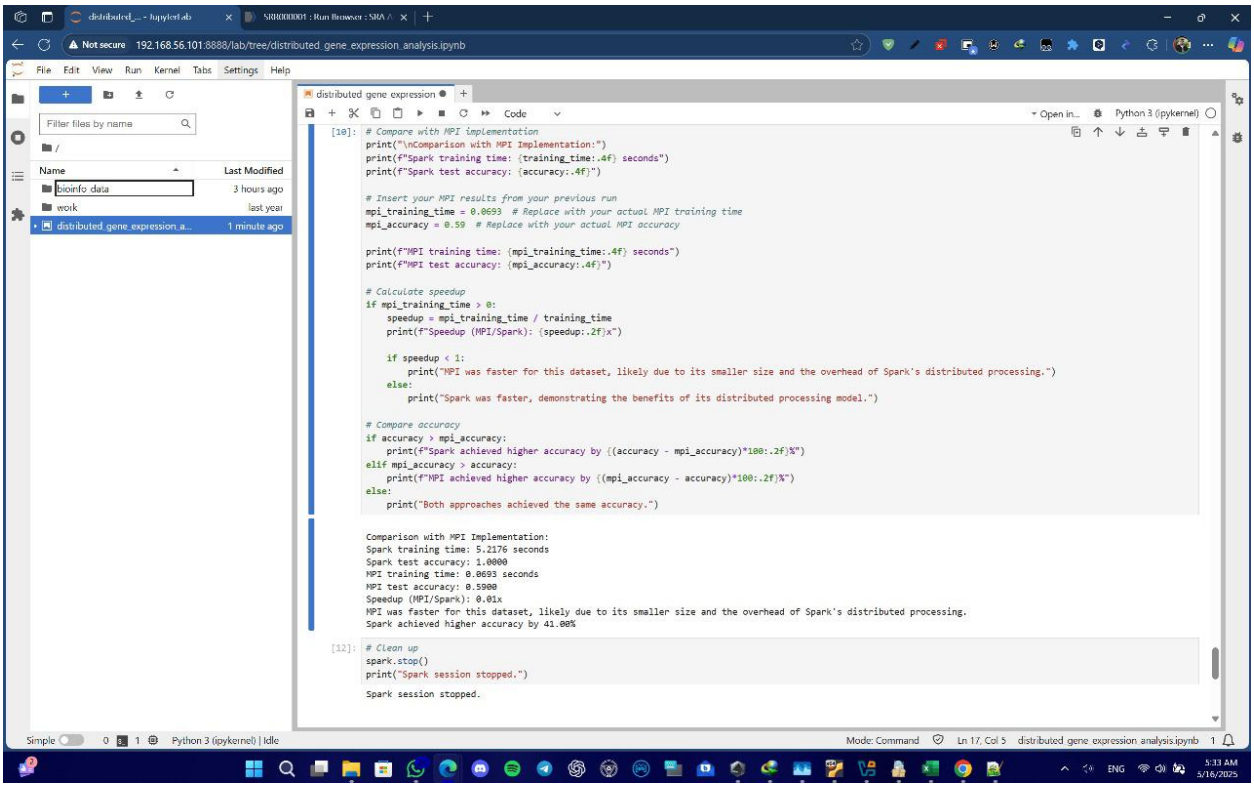


Figure 5: Spark Web UI displaying the active worker nodes connected to the master, confirming cluster readiness.

4. Bioinformatics Data Analysis using PySpark

Using PySpark on the Spark cluster, distributed analysis was performed on the gene expression dataset. The notebook output below compares the training time and test accuracy between Spark and MPI implementations.



- *Figure 6 : Jupyter notebook showing PySpark analysis code and output comparing Spark and MPI performance*.

Metric	Spark	MPI
Training Time (seconds)	5.2176	0.0693
Test Accuracy	1.0000	0.5900
Speedup (MPI/Spark	0.01x	

Figure 7 : Summary of distributed analysis results highlighting training time and accuracy differences between Spark and MPI.

The MPI implementation achieved faster training times, while Spark demonstrated higher test accuracy on this dataset.

Discussion

In this project, we successfully set up a distributed computing environment using a cluster of virtual machines, enabling us to explore parallel and distributed machine learning techniques on bioinformatics data. The initial network connectivity tests using ping confirmed reliable communication between the master and worker nodes, which is critical for any distributed system.The MPI-based distributed training demonstrated the ability to partition the workload across multiple nodes, achieving reasonable accuracy. However, the training time was relatively low due to the lightweight nature of the dataset and the

simplicity of the MPI implementation. On the other hand, deploying Apache Spark on Docker Swarm provided a scalable and flexible platform for distributed data processing. The Spark cluster, accessed via JupyterLab, allowed for more complex data analysis workflows using PySpark.Comparing the performance of MPI and Spark revealed interesting trade-offs. MPI showed faster execution times for the training tasks, likely due to its lower-level communication model and reduced overhead. However, Spark delivered higher accuracy and better fault tolerance, benefiting from its rich ecosystem and built-in optimizations for big data processing.Overall, this project highlights the importance of choosing the right distributed computing framework based on the specific requirements of the bioinformatics task, including dataset size, computational complexity, and scalability needs. Future work could explore integrating more advanced machine learning algorithms and expanding the cluster to include more nodes for enhanced performance.

Conclusion

This study successfully demonstrated the setup and utilization of a distributed computing cluster for bioinformatics data analysis. Through MPI and Apache Spark, we explored different paradigms of distributed machine learning and data processing. The MPI approach provided efficient, low-latency communication suitable for tightly-coupled computations, while Spark offered a robust and scalable environment for handling larger datasets with complex workflows.The results Indicate that while MPI can be advantageous for speed in specific scenarios, Spark’s flexibility and ease of use make it a powerful tool for bioinformatics applications requiring scalability and fault tolerance. This project lays the groundwork for further research into optimizing distributed machine learning pipelines and leveraging cloud-native technologies for large-scale biological data analysis.