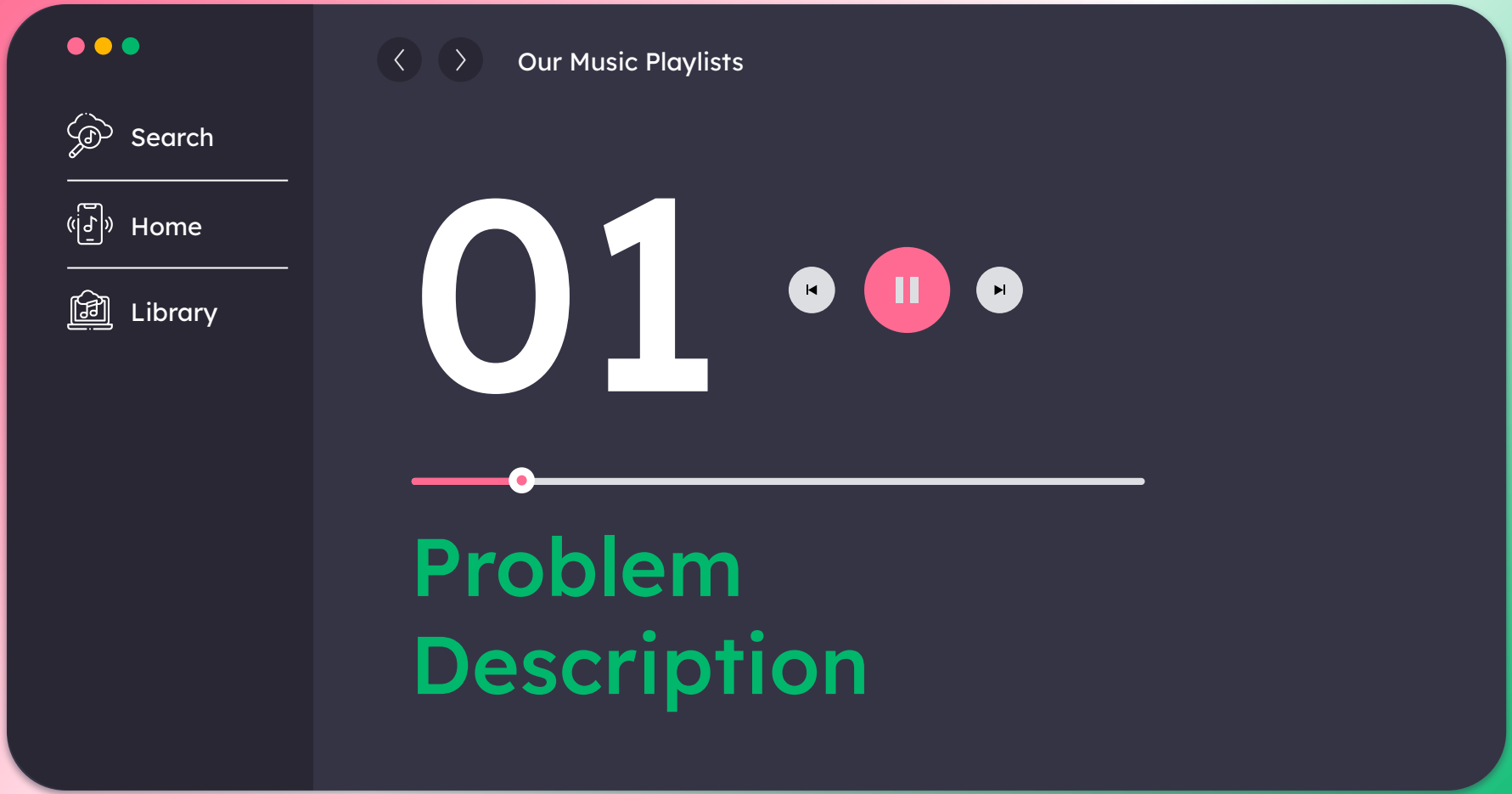

Search
Home
Library
Our Final Project
Cloud Computing Project:
Playing Federated Learning with Docker
Angelo Mandara
Claudiu Gheorghiu
Tito Tamburini


Search
Home
Library
Our Music Playlists
01
Problem
Description

Master Slave Architecture

Master-Slave Architecture: Our study utilizes a master-slave architecture for federated learning. The master node oversees the process, handling logistic regression model weights and aggregating slave nodes' updates.

Obstacles: Traditional machine learning faces challenges in era-based song classification, particularly concerning data privacy and security due to sensitive music data.

Federated Learning: Federated learning, a decentralized approach, provides the solution. It allows multiple parties to collaboratively build models while ensuring data privacy and regulatory compliance, ideal for handling sensitive song data securely.

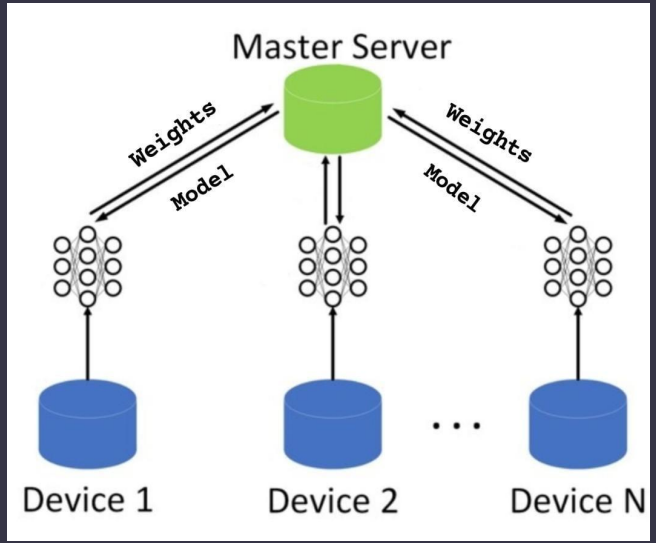
Central Challenge: Our primary challenge is classifying songs into historical eras, crucial for applications like music recommendation, historical analysis, and content curation.

Introduction to the Problem and Federated Learning

Data Security: Slave nodes independently process their data subsets, maintaining data privacy and security throughout the model-building process.

Docker for Weight Distribution: Docker efficiently transfers model weights to client nodes while enhancing data privacy through encapsulation and isolation, preventing data leaks.

Cloud Deployment: Utilizing cloud environments like AWS ensures scalability, accessibility, and secure collaboration among client nodes, meeting stringent data security requirements.



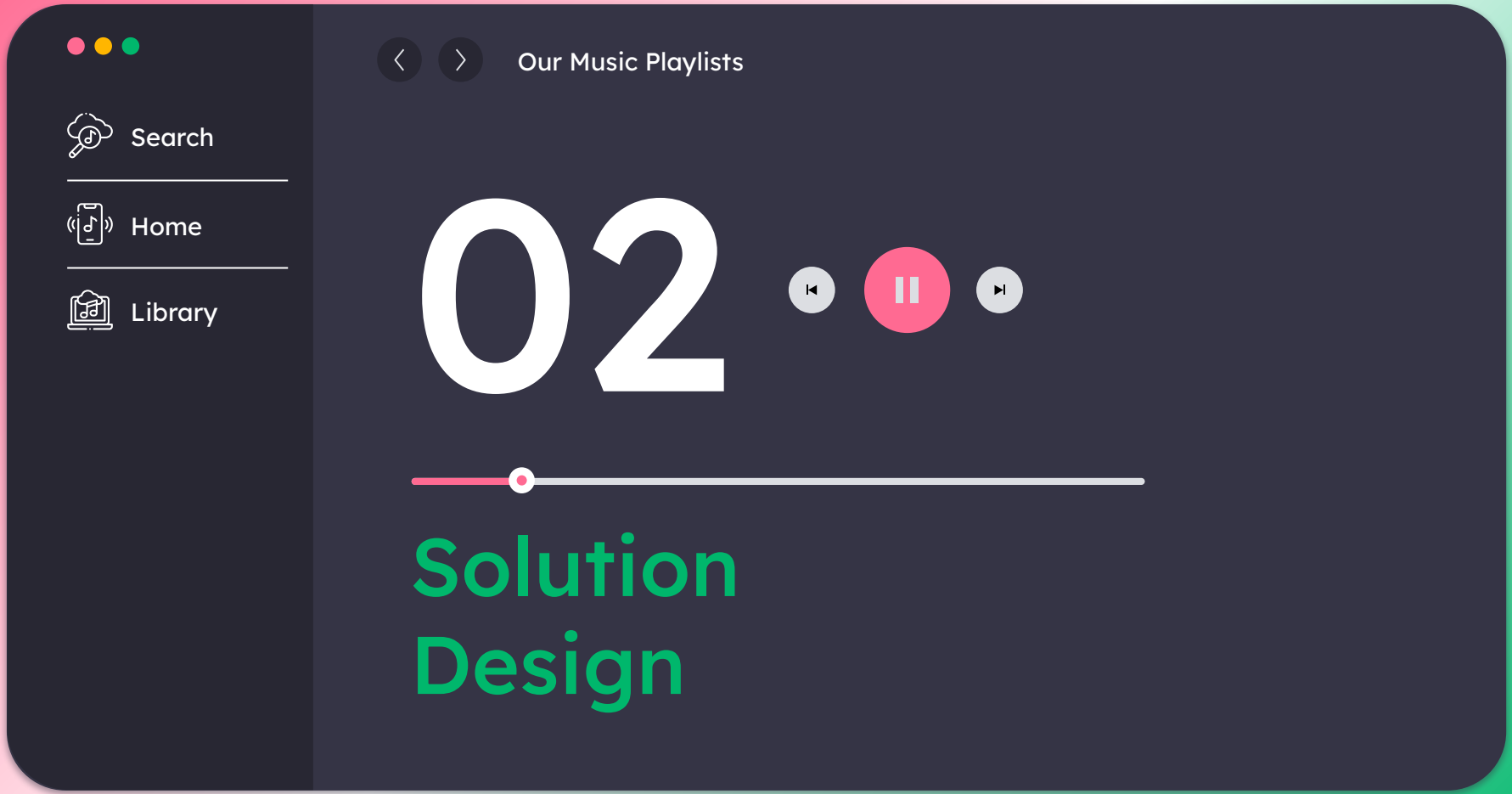
A look of the dataset

We are printing the first 10 rows of our dataset to get a quick glimpse of the data and understand its structure. Is important to say that we grouped two decades to have our final 3 classes: X-Generation(60s-70s), Y-Generation(80s-90s) and Z-Generation(00s-10s)

```
df.iloc[:,1:].head(10)
```

✓ 0.0s

	track_name	decade	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo
0	Pump Up The Jam	Y-Generation	0.885	0.844	10	-9.225	0	0.0733	0.01470	0.000004	0.0494	0.715	124.602
1	All Outta Angst	Y-Generation	0.637	0.969	0	-4.682	1	0.0432	0.00473	0.001480	0.0428	0.940	100.260
2	Walkin' On The Sun	Y-Generation	0.735	0.974	6	-4.636	1	0.0318	0.43000	0.000000	0.1450	0.967	123.290
3	54	Y-Generation	0.793	0.681	6	-6.562	0	0.0477	0.02150	0.000612	0.0863	0.792	111.011
4	Beautiful Day	Y-Generation	0.539	0.926	2	-6.495	1	0.0499	0.01400	0.001360	0.3600	0.454	136.279
5	Show Me the Meaning of Being Lonely	Y-Generation	0.630	0.625	6	-5.088	0	0.0252	0.23100	0.000000	0.0765	0.683	167.998
6	Dil Hai Ke Manta Nahin	Y-Generation	0.499	0.534	8	-9.081	1	0.0316	0.87700	0.002820	0.1950	0.717	117.524
7	Sunny Came Home	Y-Generation	0.554	0.566	11	-8.050	0	0.0332	0.33000	0.010000	0.0943	0.418	167.880
8	Pichakappoomkaavukalkkum	Y-Generation	0.788	0.844	0	-5.179	1	0.2630	0.68200	0.000080	0.1510	0.917	81.700
9	Humpin' Around	Y-Generation	0.710	0.800	1	-7.141	1	0.0400	0.00223	0.211000	0.0455	0.610	110.211


Search
Home
Library
Our Music Playlists
02
Solution
Design

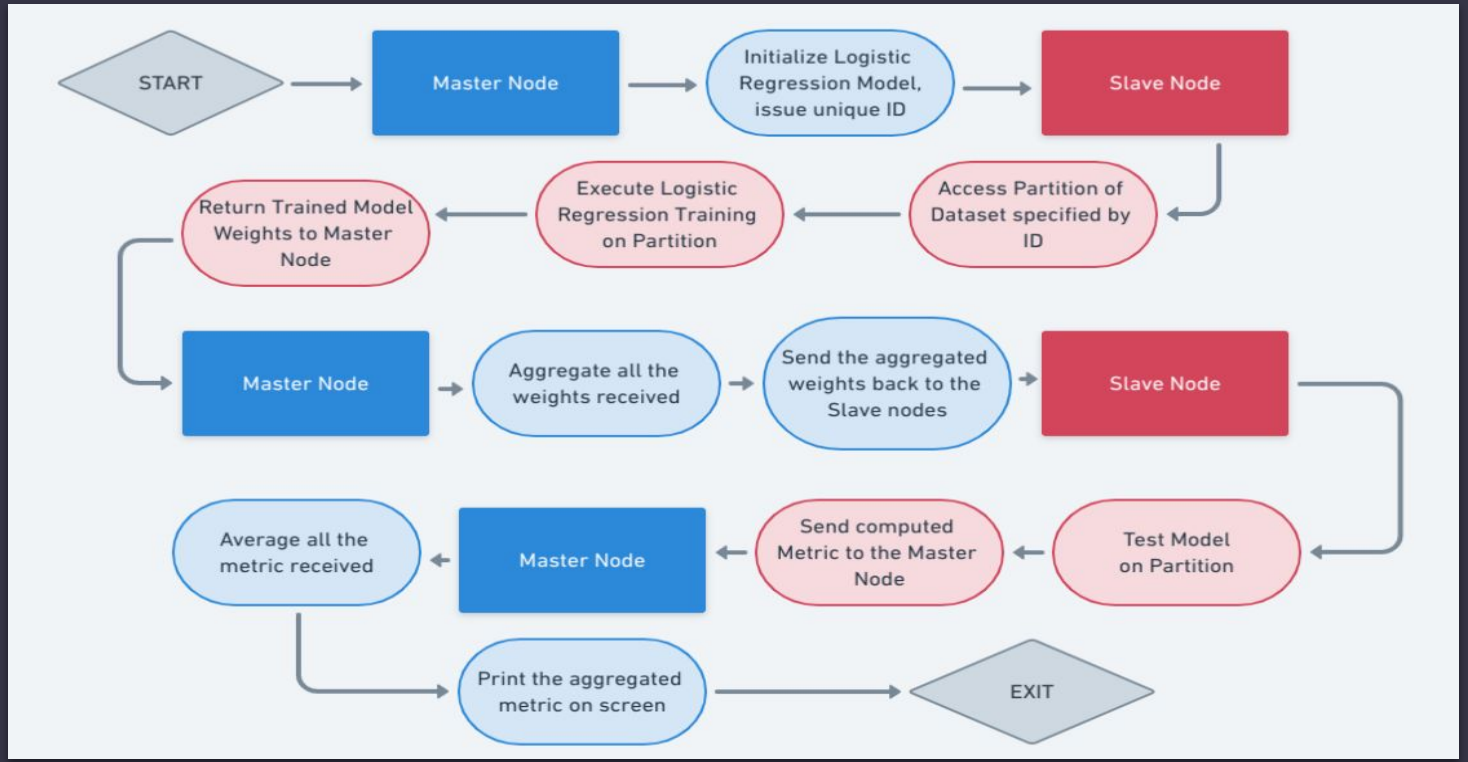
Architecture Overview

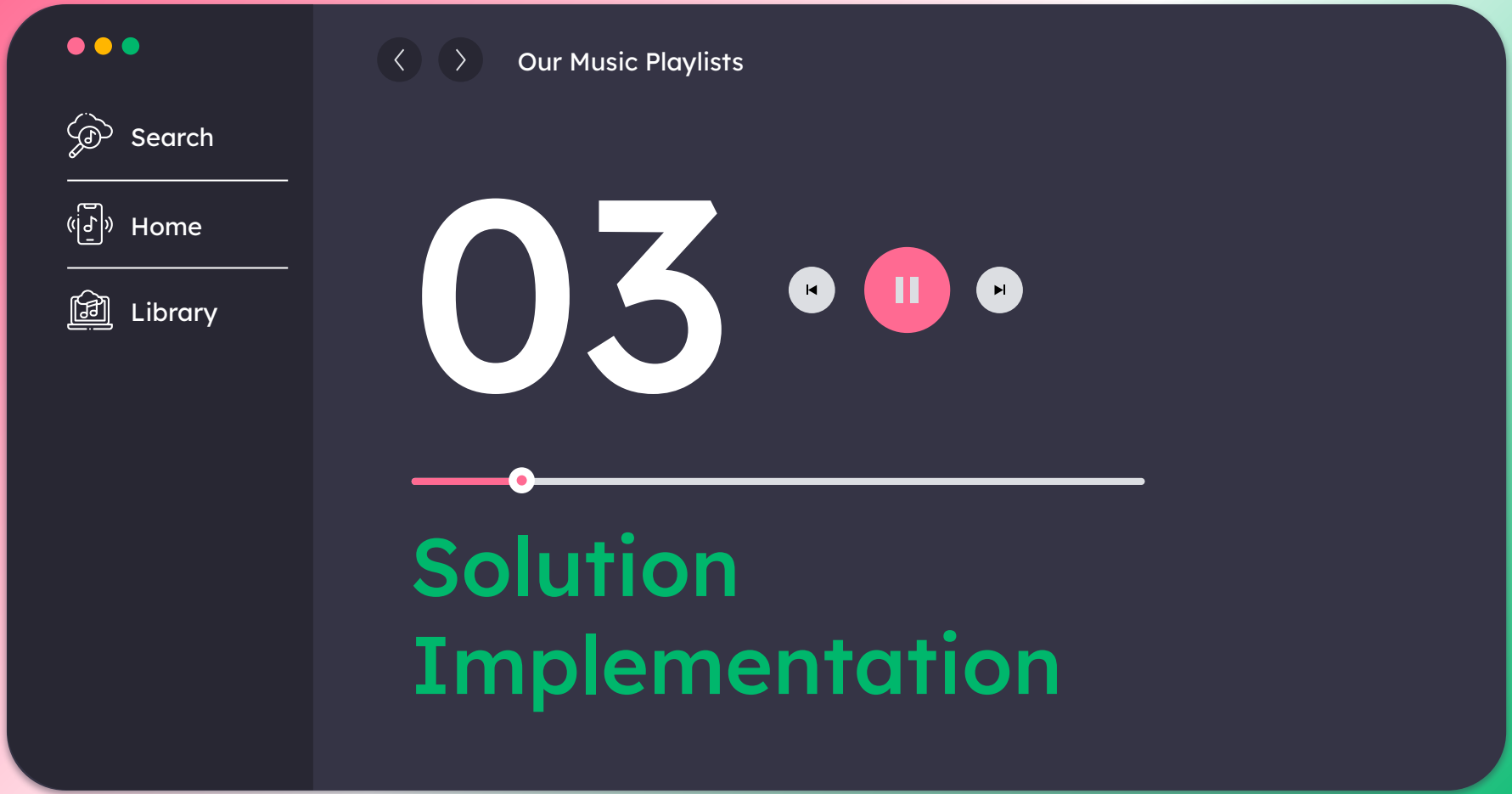
Master node coordinates the federated learning process.

- Master sends initial model weights and unique IDs to each slave node.
- Master initiates the process by initializing a logistic regression model and ensures all slave nodes start with the same initial model.
- Slave nodes access dataset partitions specified by their IDs.
- Slave nodes perform logistic regression training on their partitions.
- Slave nodes return trained model weights to the master.
- Master aggregates weights to obtain the most recent model and sent back to slaves.
- Slave nodes test the new model on their dataset divisions.
- Slave compute a metric and submit it to the master.
- Master node computes the average metric when all slave nodes submit their results.
- The federated learning program prints the final model measure on the screen.

With the optimized model, the master node assigns songs to their respective decades.

Architecture Overview




Search
Home
Library
Our Music Playlists
03
Solution
Implementation



Docker Containers: Docker is used to create isolated environments for both the master and slave nodes, ensuring reproducibility and scalability.

Communication:

- To enable communication between master and slave nodes, a well-defined RESTful API is established using Flask for the master node.
- The slave nodes send HTTP requests to the master node's API to retrieve instructions and submit results.

Master Node Implementation:

- "/get model", method = 'GET': Slave requests initialized or updated Logistic Regression Model, receives unique ID assigned by master.
- "/get model", method = 'POST': Slave sends trained model coefficients to the masterfor aggregation.
- "check server", method = 'GET': Slave checks if it can request the updated model, ensuring synchronization.
- "test model", method = 'POST': Slave sends test metrics for averaging.

Technical Details:

- Locking mechanism using Python object Mutex to prevent data collision.
- Unique IDs stored in a queue, emptied when a slave enters the Federated Learning process.
- Weights stored in Numpy arrays for aggregation.
- Counters for tracking slave progress.

Slave Node Implementation:

- Python client script.
- Communicates with the master through RESTful API services.

Data Handling:

- Pandas module for loading and reading CSV partitions.
- Custom Logistic Regression Model built from scratch to ensure consistency.

Benefits:

Privacy-Preserving Scalability Consistency Accurate Classification

Search

Home

Library

Our Music Playlists

04

Deployment of Solution

Docker Compose

Deployment Process:

- Use of AWS EC2 instances for Federated Learning application.
- Script to set the number of slave nodes and scale partitions.
- Docker images for Master and Slave nodes.
- Dataset partitioning.

```
#!/bin/bash
export NUM_SLAVES=10
cd Master
docker build -t master_image .
python3 create_partitions.py
cd Slave
docker build -t slave_image .
cd ..
docker-compose up -d --scale slave=$NUM_SLAVES
```

Deploying Federated Learning on AWS EC2 Instances

Deployment Process: Finally it deploys the containers via Docker Compose

```yaml
version: '3'
services:
  master:
    image: master_image
    networks:
      - fed_network
    environment:
      - NUM_SLAVES=${NUM_SLAVES}

  slave:
    image: slave_image
    networks:
      - fed_network
    depends_on:
      - master


networks:
  fed_network:
    driver: bridge
```

AWS EC2 Configuration

We launched multiple the same EC2 istance, with these characteristics:

- Amazon Machine Image: Amazon Linux
- Instance Type: t2.large.
- Security Group with specific inbound rules.
- User Data script for initialization.

```
#!/bin/bash
sudo yum -y update
sudo yum -y install docker python pip
service docker start
usermod -a -G docker ec2-user
chkconfig docker on
```

Application Setup:

- Install required dependencies: docker-compose and pandas.
- Download the application files an S3 bucket.
- Configure the number of slave nodes.
- Execute the application.

Search

Home

Library

Our Music Playlists

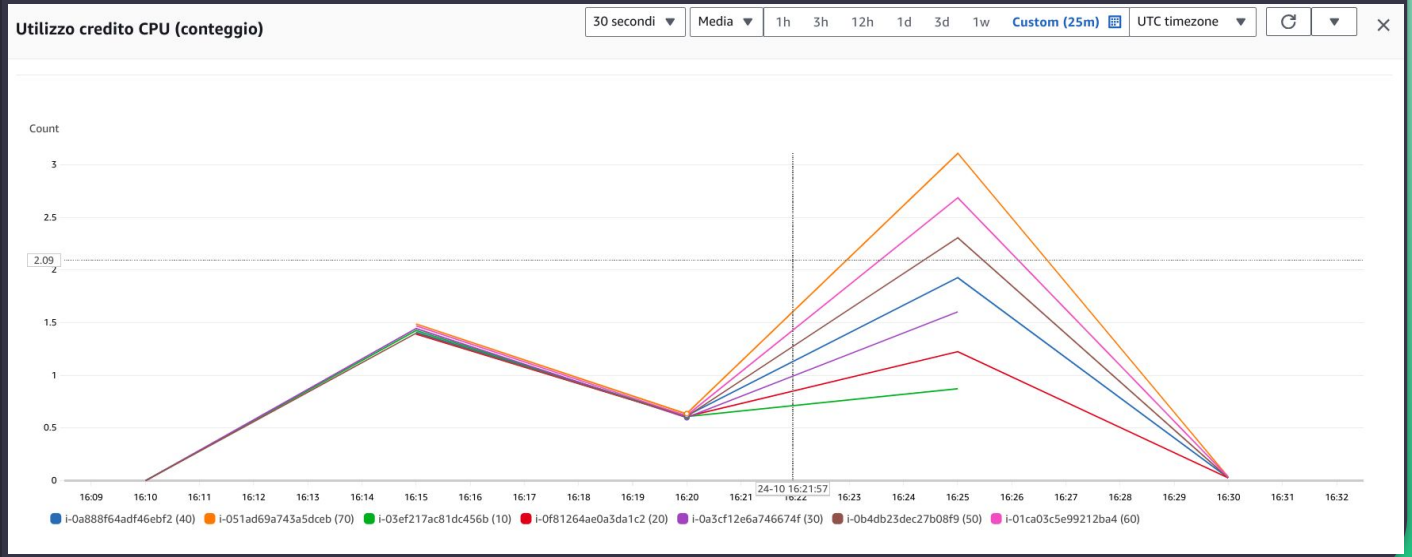
05

Experimental Design and Solution

AWS CloudWatch Monitoring and Scalability

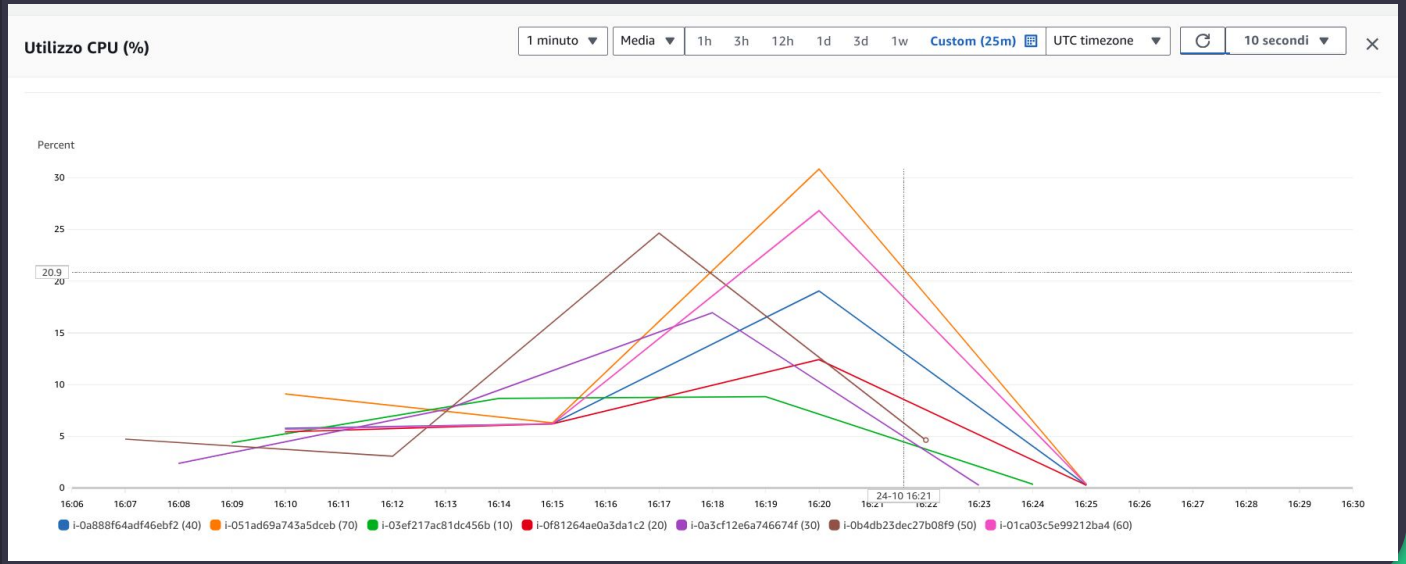
AWS CloudWatch Monitoring:

- Monitored CPU Credit Usage and CPU Usage throughout the application's execution.
- Data collected from the initialization of EC2 instances to application termination.



CPU Usage and Scalability:

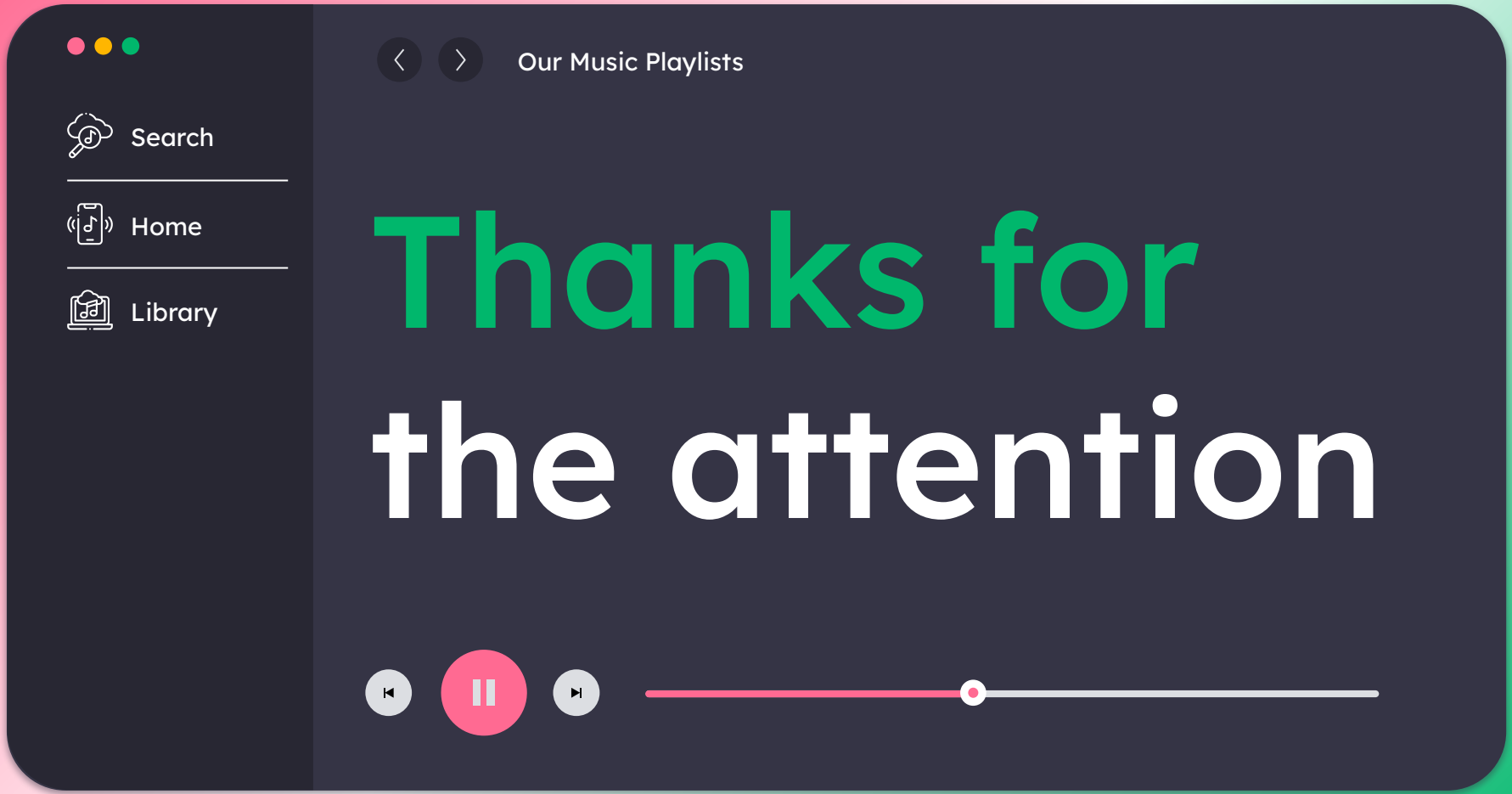
- CPU Usage for EC2 instances:
- 10 nodes (green line) reached a maximum of 10% CPU usage.
- 70 nodes (orange line) achieved a maximum of 30% CPU usage.
- The application optimally utilizes computational resources.
- Shows potential for scaling with increased nodes, indicating excellent scalability.



Future Developments

Plans for a secondary web service in the master node:

- Allow users to utilize the Federated Learning model.
- User input: Song name from Spotify.
- Model output: Predicted decade of the song's belonging.
- Expected accuracy of 66% based on training.
- Application demonstrates efficient use of AWS resources.
- Scalable for varying numbers of nodes.
- Future expansion to provide song decade predictions for users.


Search
Home
Library
Our Music Playlists
Thanks for
the attention