Project Proposal: Cloud-Based Distributed Weather Forecasting System Using Real-Time Data Streams

Team Members: Samir Sanyal, Vraj Parekh, Rajat Sawant, Dev Patel

Team Members

Name	**IU Email Address**	**Role**
Samir Sanyal	sasanyal@iu.edu	Cloud Infrastructure & Data Streaming Setup
Vraj Parekh	vrparekh@iu.edu	Data Collection & Preprocessing
Rajat Sawant	rsawant@iu.edu	Visualization Dashboard & Alerts Integration
Dev Patel	patedevj@iu.edu	Machine Learning Model Development

Problem Statement and Background

Real-time weather forecasting is critical for disaster management, agriculture, transportation, and public safety. With the increasing availability of live data from IoT devices, satellite feeds, and online APIs, traditional forecasting systems often struggle to handle the scale, velocity, and complexity of modern weather data. These legacy systems are typically centralized, lack scalability, and are not optimized for real-time processing, leading to delays in delivering accurate forecasts and alerts.

To address these challenges, our project proposes a **scalable, distributed cloud-based system** capable of handling large-scale, real-time weather data ingestion, processing, and forecasting. By leveraging cloud-native services, parallel processing frameworks, and machine learning models, our system aims to deliver accurate, low-latency weather predictions and real-time alerts for extreme weather conditions.

Key References

- **NOAA Open Weather Data APIs**: A reliable source of real-time and historical weather data.
- **Google Cloud Weather Dataset**: A comprehensive dataset for weather analysis and forecasting.
- **Research on LSTM-based Weather Prediction Models**: Studies demonstrating the effectiveness of Long Short-Term Memory (LSTM) networks for time-series weather forecasting.
- **Apache Spark Streaming Documentation**: Resources for implementing distributed data processing pipelines.

- **AWS SageMaker and Google AI Platform**: Cloud-based platforms for training and deploying machine learning models.

Specific Goals and Expected Impact

Goals

1. **Real-Time Data Ingestion Pipeline**: Build a scalable, real-time data ingestion pipeline for weather data using cloud-native tools like **Apache Kafka**, **AWS Kinesis**, or **Google Pub/Sub**. This pipeline will handle high-velocity data streams from multiple sources, including NOAA APIs, satellite feeds, and IoT sensors.

2. **Parallel Data Processing**: Implement a distributed data processing framework using **Apache Spark Streaming** or **Google Dataflow** to preprocess, aggregate, and analyze real-time weather data across multiple cloud instances. This will ensure efficient handling of large datasets and reduce processing latency.

3. **Machine Learning for Weather Forecasting**: Develop and deploy a machine learning model (e.g., **LSTM-based neural network**) for time-series forecasting of weather conditions. The model will be trained on historical weather data and deployed using cloud platforms like **AWS SageMaker** or **Google AI Platform**.

4. **Interactive Visualization Dashboard**: Design an interactive, real-time dashboard using tools like **Grafana** or **Streamlit** to display weather forecasts, trends, and extreme weather alerts. The dashboard will provide users with actionable insights and visualizations of weather data.

5. **Real-Time Alert System**: Set up a real-time alert system to notify users of severe weather conditions (e.g., storms, floods, or heatwaves) through **SMS**, **emails**, or **push notifications**. This system will leverage serverless cloud functions like **AWS Lambda** or **Google Cloud Functions** for efficient and scalable alert delivery.

Expected Impact

- **Improved Disaster Response**: By providing real-time weather forecasts and alerts, our system can help disaster management agencies respond more quickly to extreme weather events, potentially saving lives and reducing property damage.

- **Enhanced Public Safety**: The system will enable proactive measures for public safety, such as early warnings for severe weather conditions.

- **Agricultural Benefits**: Farmers can use the system to make informed decisions about planting, irrigation, and harvesting based on accurate weather predictions.

- **Scalability and Efficiency**: The cloud-based architecture ensures that the system can scale to handle increasing data volumes and user demands, making it suitable for global deployment.

Methodology and Required Resources

Methodology

1. **Data Collection**:
    - Integrate real-time weather data streams from public APIs (e.g., NOAA), satellite feeds, and IoT weather sensors.
    - Ensure data quality and consistency by implementing preprocessing steps such as data cleaning, normalization, and deduplication.
2. **Distributed Processing**:
    - Use **Apache Spark Streaming** or **Google Dataflow** for parallel processing of weather data across multiple cloud instances.
    - Implement data aggregation, filtering, and transformation to prepare the data for machine learning and visualization.
3. **Forecasting Model**:
    - Train an **LSTM-based neural network** using historical weather data to predict future weather conditions.
    - Deploy the trained model on a cloud platform (e.g., AWS SageMaker or Google AI Platform) for real-time inference.
4. **Alert System**:
    - Utilize serverless cloud functions (e.g., **AWS Lambda** or **Google Cloud Functions**) to trigger notifications during extreme weather events.
    - Integrate with third-party services (e.g., Twilio for SMS or SendGrid for emails) to deliver alerts to users.
5. **Visualization**:
    - Build a live dashboard using **Grafana** or **Streamlit** to display real-time weather data, forecasts, and alert statuses.
    - Ensure the dashboard is user-friendly and provides actionable insights for end-users.

Required Resources

- **Cloud Platform**: AWS or Google Cloud Platform (final selection pending feasibility tests).
- **Compute Resources**: GPU-enabled instances for model training and Kubernetes clusters for scalable deployment.
- **Frameworks**: Apache Kafka, Apache Spark, TensorFlow/PyTorch, Grafana/Streamlit.
- **APIs/Datasets**: NOAA Weather Data, Google Cloud Weather datasets.
- **Alert Services**: Twilio (for SMS), SendGrid (for emails), or Firebase (for push notifications).

Timeline

Week	Milestone
Week 1-2	Finalize system architecture and set up data ingestion pipelines.
Week 3-4	Implement parallel data processing framework (Spark/Dataflow).
Week 5-6	Develop and train the LSTM-based weather forecasting model.
Week 7-8	Integrate the real-time alert system using serverless cloud functions.
Week 9-10	Build and refine the interactive visualization dashboard.
Week 11	Conduct performance optimization, testing, and debugging.
Week 12	Prepare the final report and presentation.