DESIGN DOCUMENT

ABSTRACT

The project aims to develop a distributed machine learning inference model for real-time trajectory predictions. The model will utilize a distributed computing architecture to enable efficient processing of large-scale data and provide accurate trajectory predictions in real-time. The model will be designed and implemented using state-of-the-art machine learning techniques, and trained on a dataset of trajectory data using supervised learning algorithms. The project will involve designing and implementing algorithms for data distribution, load balancing, and fault tolerance to ensure the model's robustness and scalability. The performance of the model will be evaluated using metrics such as prediction accuracy, response time, and resource utilization. The ultimate goal of the project is to develop a highly performant and scalable distributed machine learning inference model that can be deployed in various real-world scenarios, such as autonomous driving, logistics, and transportation, where real-time trajectory prediction is critical for decision-making and safety.

PROJECT DESIGN

The system design of the project includes the following components:

1. **Data Ingestion:** The system ingests trajectory data from various sources, such as GPS-enabled devices and traffic sensors. The data is pre-processed and formatted for ingestion into the distributed machine learning model.
2. **Data Storage:** The pre-processed data is stored in a distributed storage system, such as Hadoop Distributed File System (HDFS), to enable efficient data access and processing.
3. **Data Distribution:** The distributed machine learning model distributes the data across multiple nodes to enable parallel processing of the data. The data is partitioned based on the workload and the computational resources available on each node.
4. **Model Training:** The model is trained using state-of-the-art machine learning techniques, such as deep learning and reinforcement learning algorithms. To reduce prediction error and raise prediction accuracy, the model's parameters are adjusted during the training phase.

5. **Model Inference:** Once the model is trained, it is deployed for real-time trajectory prediction. The model receives new trajectory data in real-time and provides predictions based on the data received.
6. **Load Balancing:** To ensure optimal resource utilization and efficient processing, the system includes load balancing algorithms that balance the computational workload across multiple nodes.
7. **Fault Tolerance:** The system includes fault tolerance mechanisms that enable the system to recover from node failures or system crashes without impacting the overall performance of the system.

The overall architecture of the project can be divided into five steps:

1. **Data Collection and Pre-processing:** Data collection involves gathering large amounts of real-time data on trajectories from various sources. This data is collected using sensors, cameras, GPS devices, and other IoT devices. Further data pre-processing ensures cleaning, transforming, and aggregating the collected data to make it suitable for training the machine learning models. This step is crucial to ensure that the data is of high quality and is representative of the real-world scenarios.
2. **Model Partitioning:** In this step, the model is split layer-wise using Keras and TensorFlow libraries to create a new DAG with the desired layers. A new model containing only the partitioned layers is then created and sent to the Compute Nodes.
3. **Configuration Step:** The partitions are configured on the compute nodes during the configuration step, and each compute node creates two TCP sockets with the dispatcher node: one for the model's weights and IP address, the other for the model's architecture. Also, each compute node starts a new thread that connects to the node that will be the next in the inference chain and provides intermediate inference results to that node.
4. **Distributed Inference:** The dispatcher transmits input to the first compute node in the chain and distributes model partitions among compute nodes during the distributed inference step. The Compute Nodes do inference on the model once deserializing and compressing the data are received. The outcome is then compressed and serialized before being transferred via the outgoing connection. Each Compute Node goes through this process once, with the sockets doing inference calculations serially and first-in-first-out.

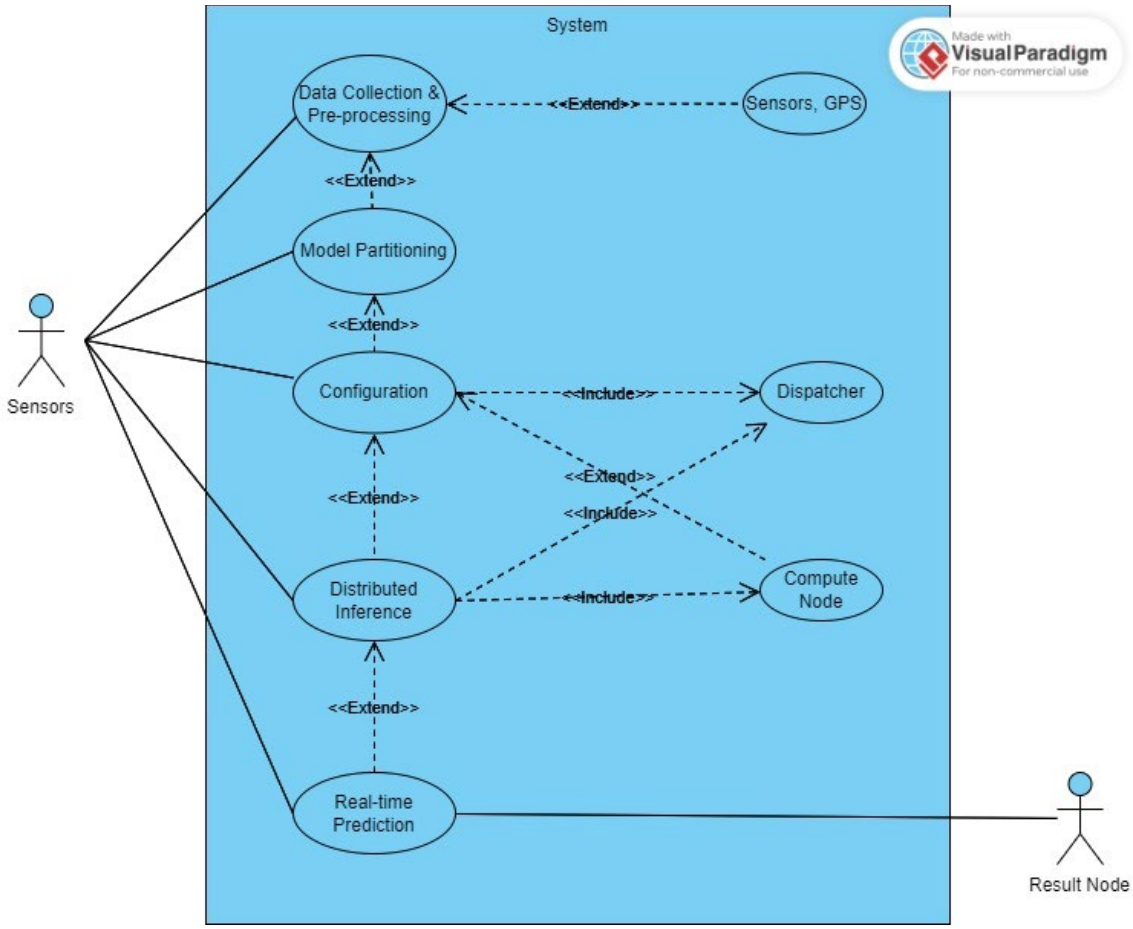
All socket connections use chunked data transfer to reduce the amount of data needed to build a model and communicate intermediate inference results while maximizing system performance.

5. **Real-time Prediction:** The final component is real-time prediction, which involves using the trained models to predict the future trajectories of objects. The prediction is made in real-time and is based on the latest sensor data.

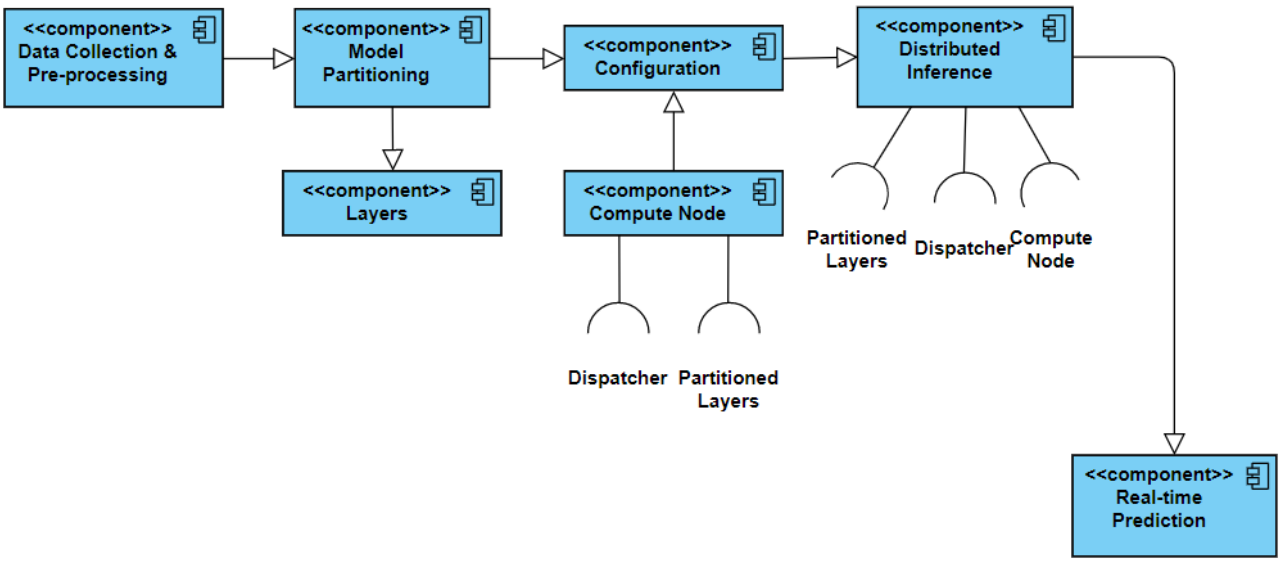
Results Generation is the final component executed in the project which involves comparison of:

- Performance (accuracy of the model) and Latency for
    - Distributed and Non – Distributed
    - Inference Throughput
    - Energy Consumption

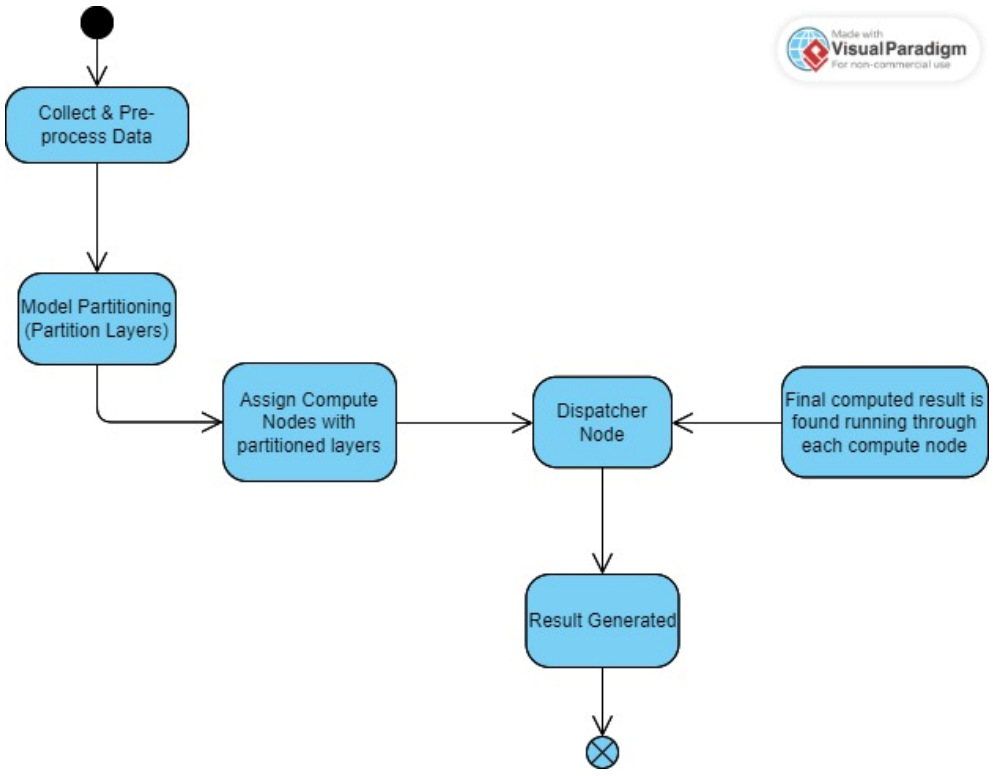
Use Case Diagram



Component Diagram



Activity Diagram



PROJECT SCHEDULE

The proposed project will take approximately 6 weeks to complete. The project timeline is as follows:

1. Week 1 (March 24) → Data Collection and Pre-processing
2. Week 2 (March 31) → Model Partitioning
3. Week 3 & 4 (April 14) →Configuration
4. Week 5 (April 21) → Distributed Inference
5. Week 6 (April 28) → Results Generation and Evaluation