Group Project Milestone #1

Distributed LLM Lifecycle Management Platform for Large-Scale Multi-Cloud and Edge Environments

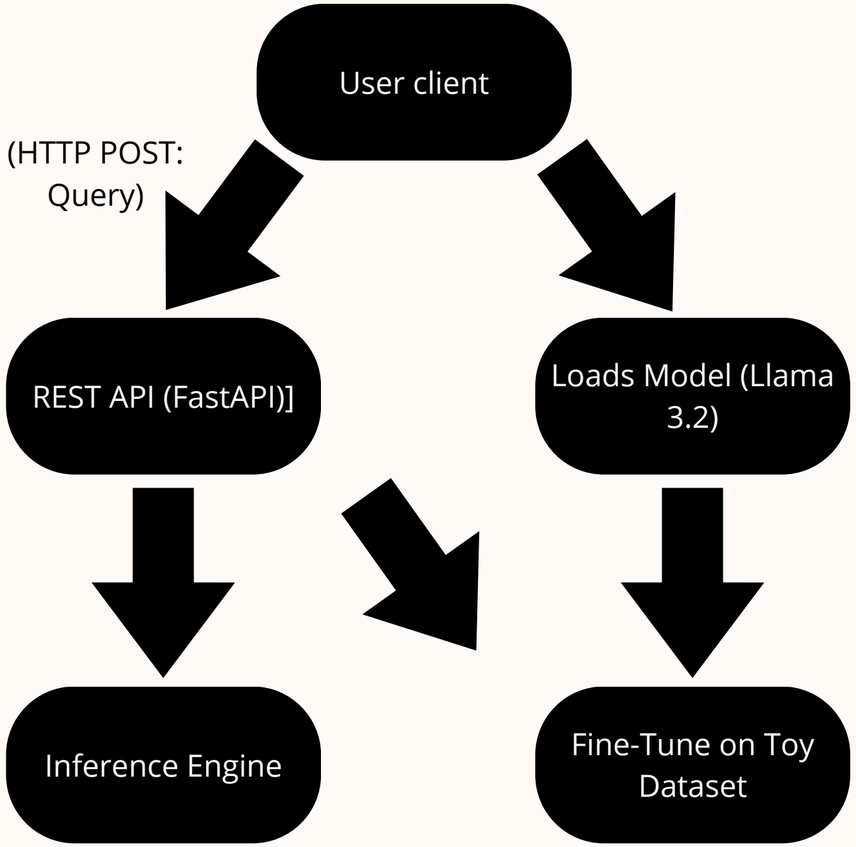
Spring 2025: Group 22: Efta Bashar / Anton Chendea / Kazi Haque

Abstract presentation and introduction

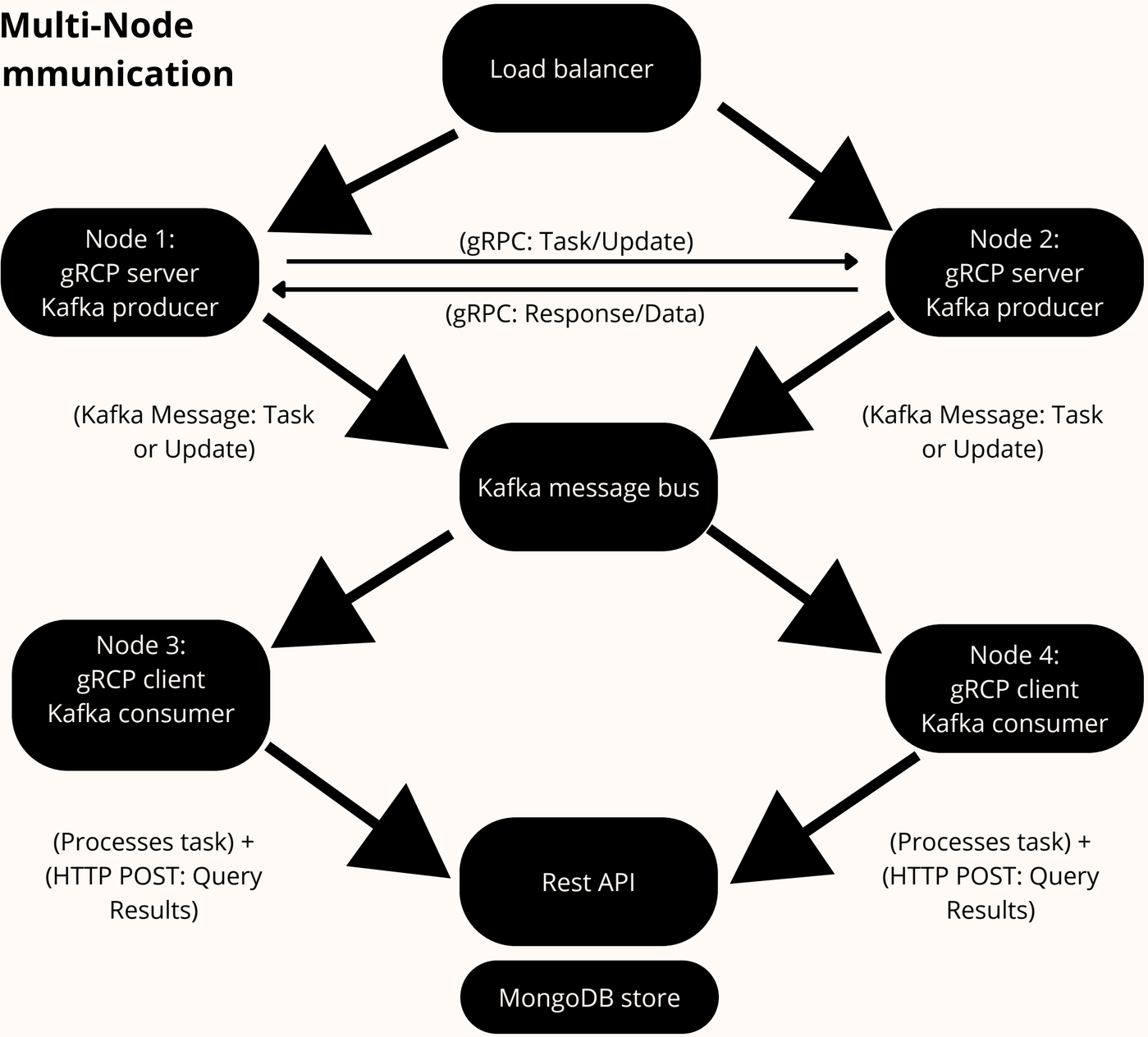
In this project, the objective is to design and implement a distributed system for training and fine-tuning a Large Language Model (LLM) interface. The system integrates multiple interconnected nodes using communication protocols like gRPC and Kafka, with FastAPI facilitating user interactions. Tools such as PyTorch and MongoDB are utilized for efficient data management and scalability, ensuring smooth operation across distributed components.

System Design Diagram

Single node prototype



Multi-Node Communication



Communication protocols

gRPC : Used for a fast and efficient communication between nodes because it’s a high-performance, low-latency communication protocol designed for distributed systems. Uses Protocol Buffers for efficient serialization and supports bidirectional streaming. It’s good for node to node interaction, perfect for model updates, inference tasks, and real-time data exchanges in distributed architectures.

Kafka : Offers fault tolerance through message replication and partitioning, ensuring reliable communication even in the event of node failures. Because it’s enables asynchronous, scalable message exchange across distributed systems. Decouples services, facilitating event-driven architectures, like distributing model updates or queuing inference tasks for later processing.

MongoDB : A flexible NoSQL database that stores unstructured data in JSON-like documents. Scalable, allowing for horizontal sharding and partitioning. Ideal for storing dynamic data, such as model configurations, training logs, and inference results. Thats provide agility in managing large, evolving datasets without predefined schemas.

Fast API : web framework which is modern and high-performance for building APIs with Python. Designed for speed, with asynchronous support and automatic data validation. FastAPI simplifies building RESTful APIs, that’s very good for creating endpoints that handle inference requests, manage models, and interact with clients..

Software, Programming Languages, and Frameworks

Languages used:

-Python for REST API, model training, or Kafka management.

-Protocol Buffers for defining gRPC messages

Frameworks:

-PyTorch/TensorFlow for loading and fine-tuning the model

-FastAPI implementation

-Kafka (message management)

Additional tools:

-Kubernetes for containerizing and managing nodes.

-MongoDB for storing data and results

Demo video

Challenges