CISC 322 AI Presentation

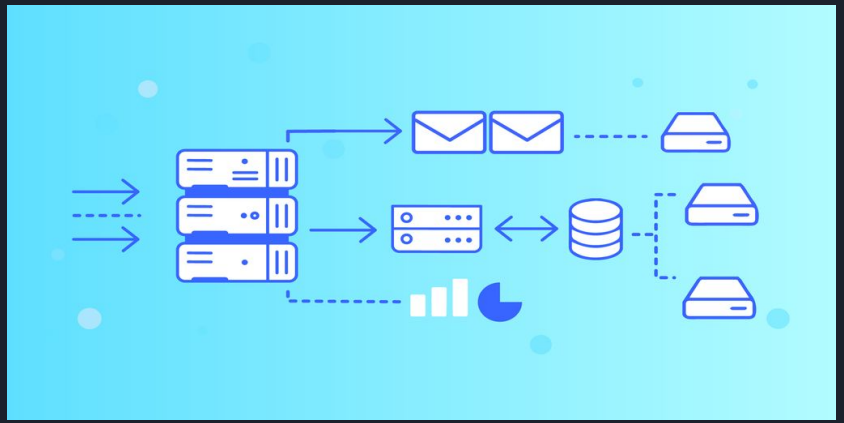
Adam Cockell, Ashton Thomas, Udbhav Balaji, Dahyun JIN, Udit Kapoor, Kevin Subagaran

Introduction 1

- Apollo is an open-source, performance-focused framework designed for autonomous driving
- Aims to be developer-friendly and flexible for a wide range of applications
- Uses the pipe-and-filter architectural style to manage data flow between modules
- Handles processing for the components using a scheduler which dynamically assigns tasks based on resources available

Introduction 2

- Apollo is composed of 5 platforms: Solutions, Cloud Service Platform, Open Software Platform, Hardware Development Platform and Open Vehicle Certification Platform
- The software platform can be further broken down into 8 modules: map engine, localization, perception, prediction, planning, control, HMI and Apollo CyberRT
- We aim to analyze the conceptual architecture behind Apollo and discuss the modules comprising the open software platform

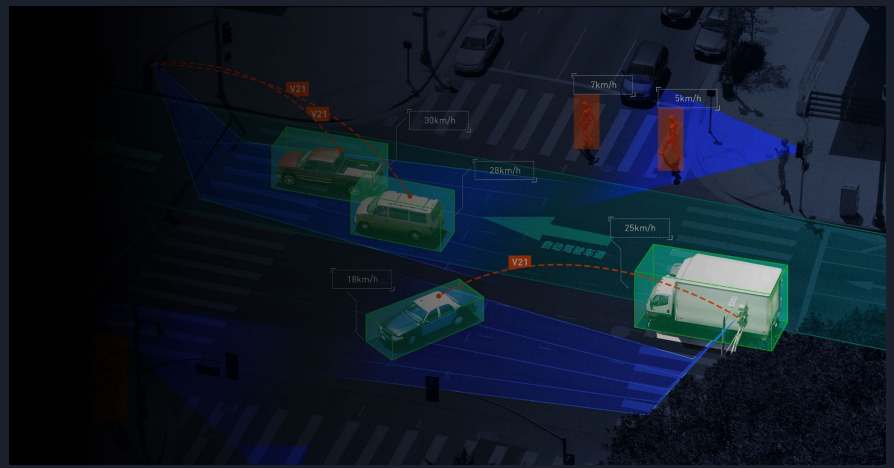


Architecture Modules - Localization

- This module is responsible for performing all the required localization services
- Requires a high level of accuracy
- RTK (Real-Time Kinematic) method, using GPS and IMU (Inertial Measurement Unit) information
- Multi-sensor fusion method, using data from GPS, IMU and LiDAR

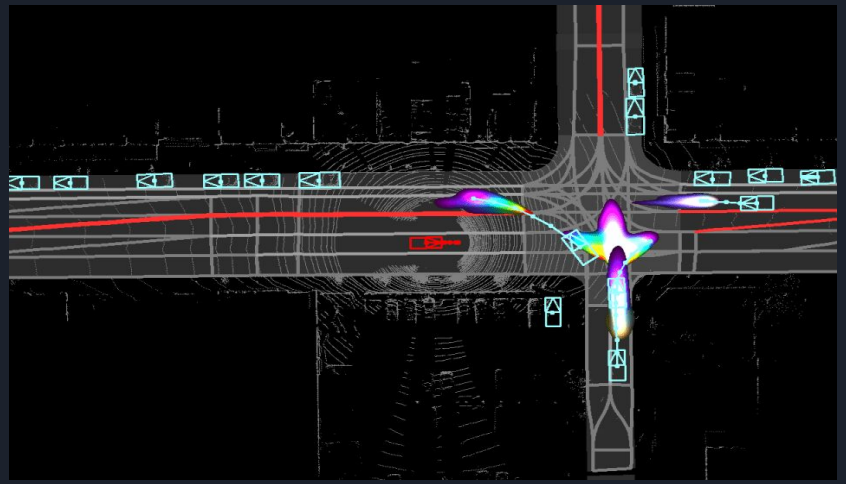
Architecture Modules - Perception

- This module is responsible for ensuring that all the obstacles in the car's path are properly detected
- Takes LiDAR data, radar data, image data, as well as other raw data
- Receives output from the Localization module for further processing and prepares it for the Prediction module



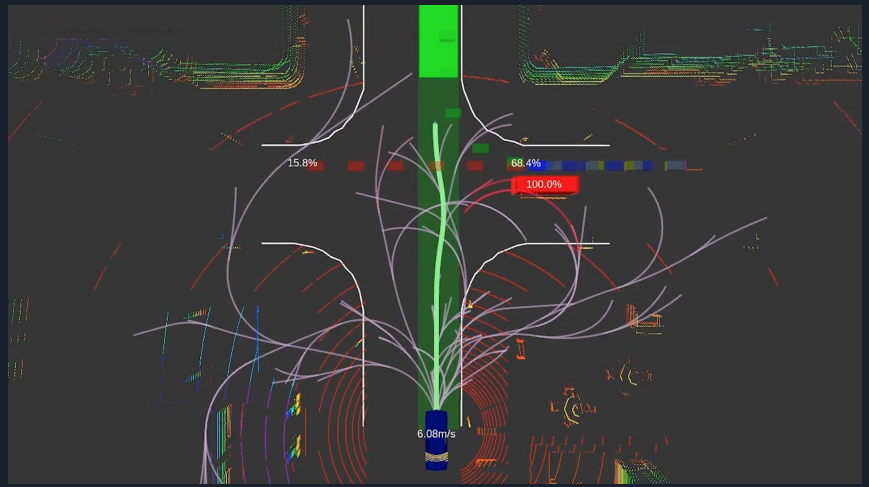
Architecture Modules - Prediction

- This module is responsible for studying and predicting the behavior of obstacles detected by the perception module
- Takes basic perception information like positions, headings, and velocities
- Receives output from the Perception and Localization modules
- Generates predicted trajectories, with risk probabilities for detected obstacles



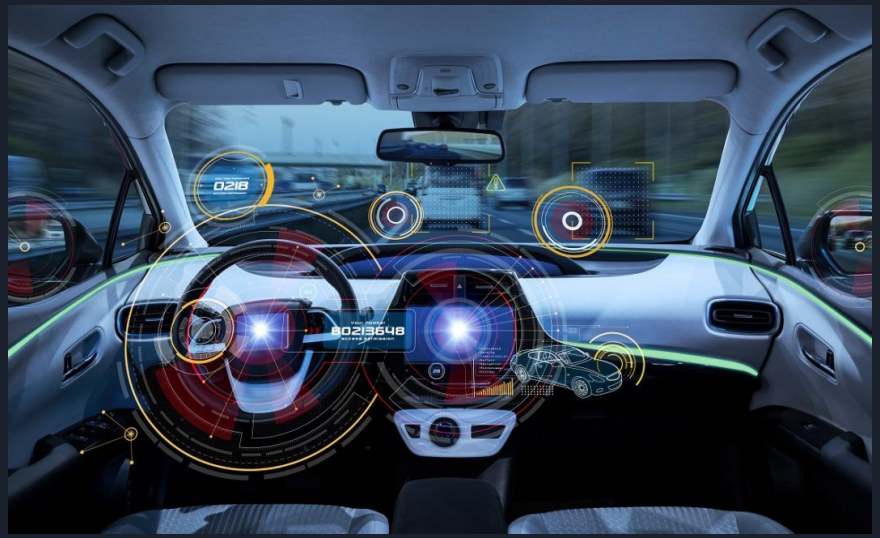
Architecture Modules - Planning

- This module is responsible for finding a collision-free and comfortable navigation path
- Receives output from the Prediction and Perception modules
- Accounts for several important scenarios such as approaching traffic lights, parking, and emergency situations while abiding by traffic laws
- Makes the ultimate decision in route before sending it to the Control module



Architecture Modules - Control

- This module issues commands to the steering wheel, throttle and brake to control the vehicle
- Performs some processing to ensure a comfortable and optimal route
- It is crucial that received output is strictly safe, legal maneuvers



Architecture Flows - Data Flow

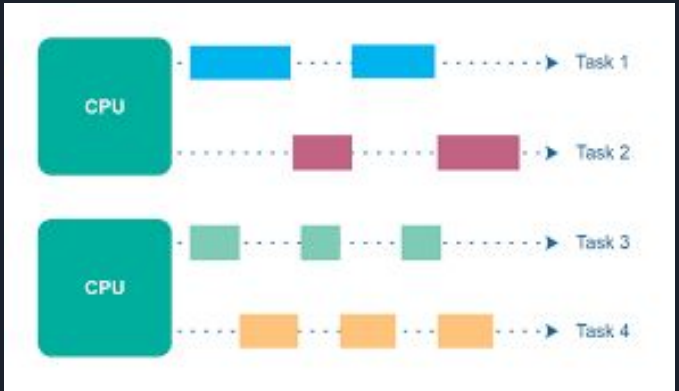
- Data is collected from the HDMap module and is sent to the Localization, Perception and Prediction modules
- Once the Prediction module finishes execution, data is sent to the Planning module
- The Localization module output is used by the Perception and Prediction modules
- The Perception module takes additional position-related data along with data from the Localization module

Architecture Flows - Control Flow

- The flow of control in the system is more linear than the flow of data
- Control flow moves linearly from the Perception module, to the Prediction module, onto the Planning and Control module

Architecture - Concurrency

- Autonomous driving requires a high level of optimization in order to minimize latency
- Apollo Cyber RT was developed as the world's first open-source runtime framework designed for development of autonomous driving technologies
- The individual components are combined with raw sensor data to create lightweight jobs which are executed via a priority queue
- All this ensures the framework works at the highest level of performance without requiring heavy and complicated deployments



Architecture - Evolution of the System

- Apollo has evolved over 10 updates in the past decade from July 4, 2017, the first release date, to December 28, 2021, the most recent update. The architecture of the Apollo framework has developed significantly in that time, while important updates are constantly being worked on. Overall, the architecture is divided into 4 different platforms: Cloud Service Platform, Open Software Platform, Reference Hardware Platform, and Reference Vehicle Platform.

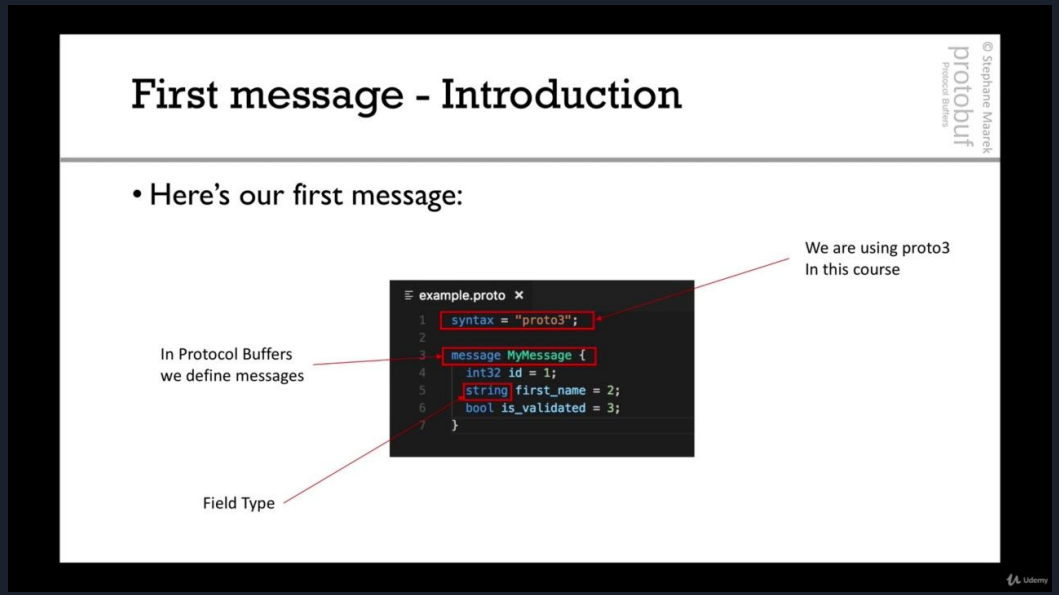
1. Enclosed venue -> Simple urban road (v1.0.0 - v2.0.0)
2. Simple urban road -> Geo-fenced highways (v.2.0.0 - v2.5.0)
3. Closed venue (v.2.5.0 - v3.0.0)
4. Complex driving scenarios (v.3.5.0 - v5.5.0)
5. Enhancing the Apollo Perception and Prediction modules (v.6.0.0 – v7.0.0)

External Interfaces - Dreamview

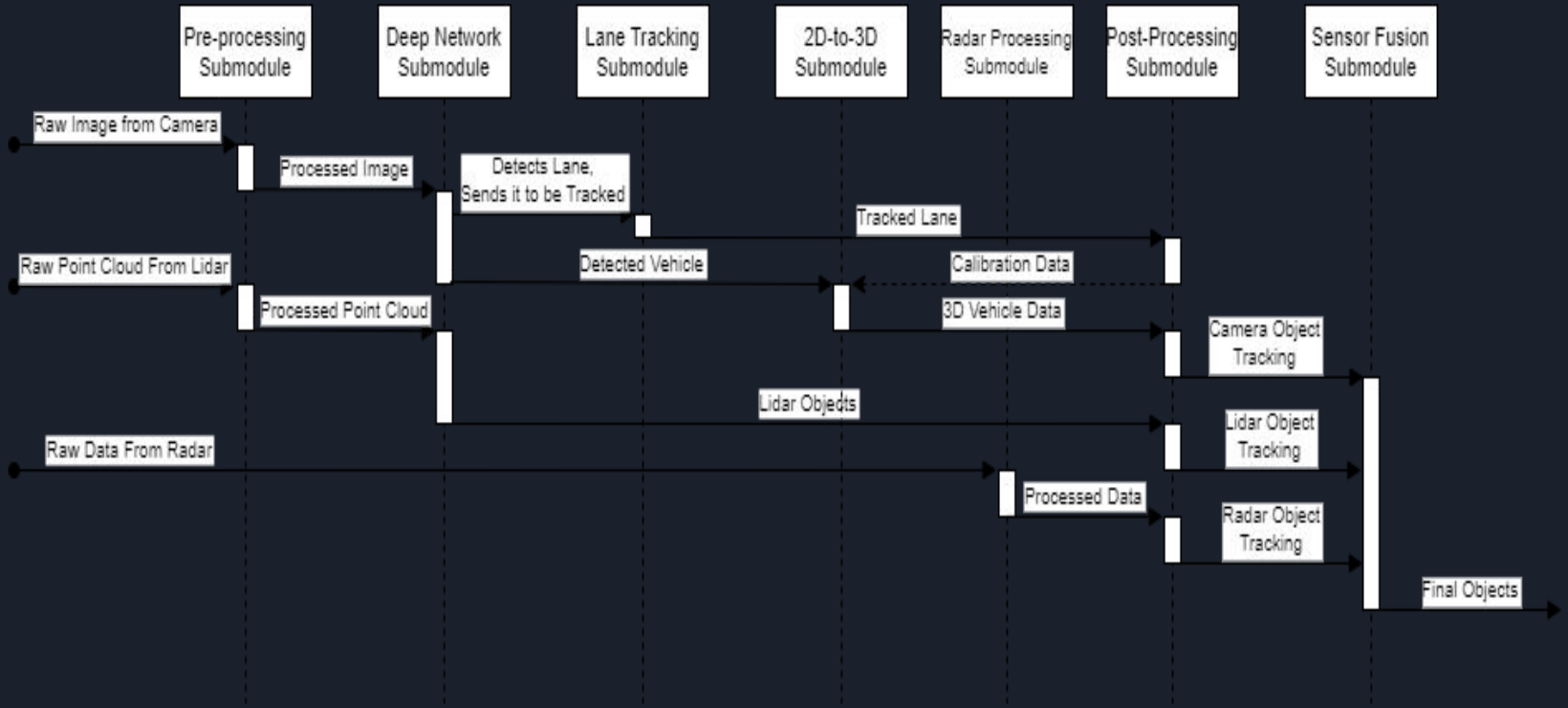
- Dreamview provides a web application that can help developers better understand the outputs of other modules.
- It is a dynamic 3D HMI designed to help developers better understand the outputs of individual components of the system.
- It can display data from the Localization, Chassis, Planning, Monitor, Perception Obstacles, Prediction, and Routing modules.
- More specifically, it can display the vehicle's planned trajectory, obstacles around the car and the chassis status among other things.

External Interfaces - Protobuf Messages

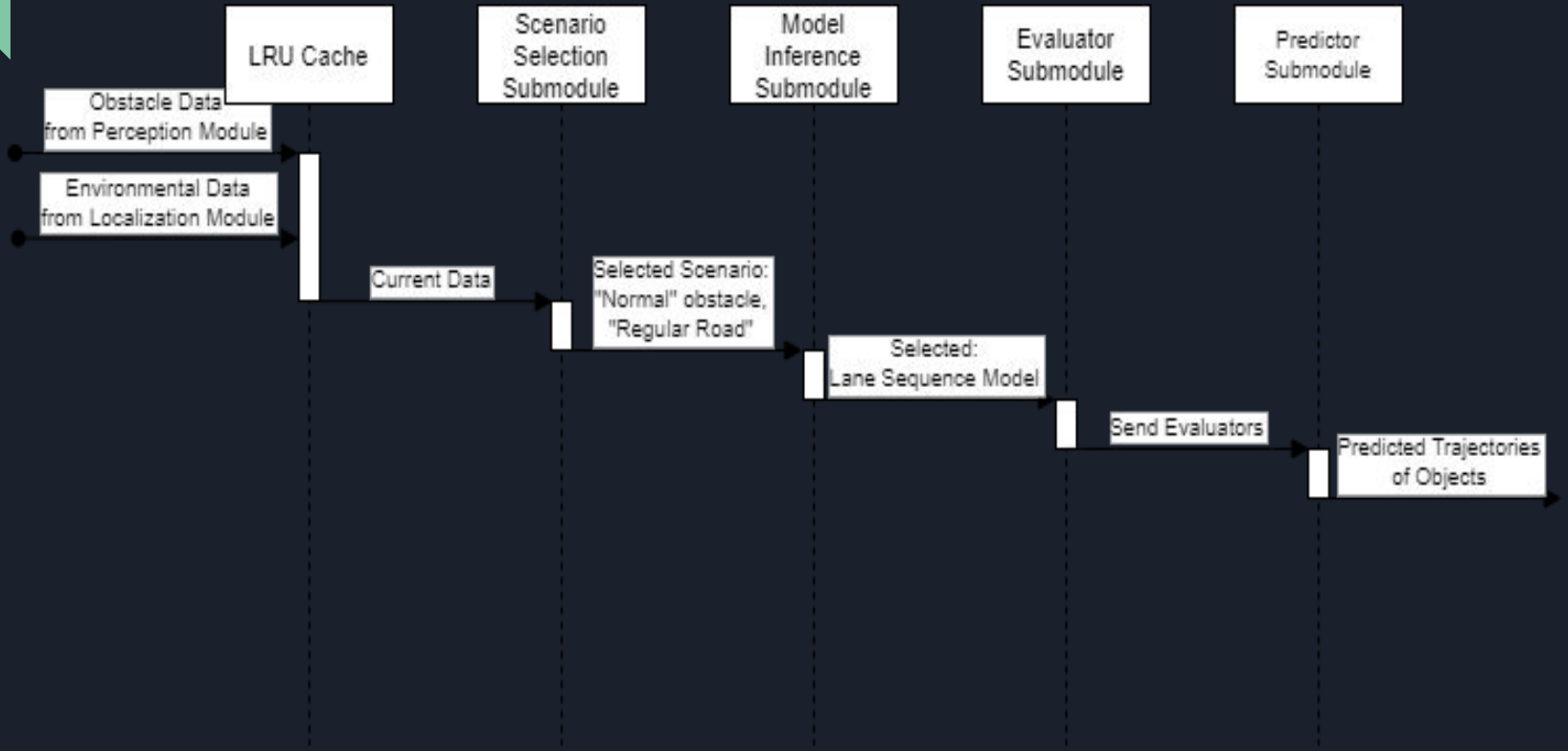
- Protobufs (Protocol Buffers) are a language and platform agnostic mechanism for sending and serializing structured data.
- It is used within Apollo to send messages between Modules and Submodules.



Use Case #1 - Perception Module



Use Case #2 - Prediction Module



Team Issues/Lessons Learned - 1

The Apolong, also known as the Baidu Apollo project, is a self-driving vehicle developed by Baidu. The company's Apollo open-source autonomous driving platform attracted 70 significant partners in 2017, including Hyundai Motors, ROS, technological startups, and others.

Therefore, it is evident that Apollo is a promising and large open-source project. The Apollo Auto GitHub mirror has gotten a lot of attention and has sparked a lot of discussion. As a result, several various methods have been put in place to guarantee that every commit is properly tested and reviewed, and that every single feature is successfully merged into the source code.

Team Issues/Lessons Learned - 2

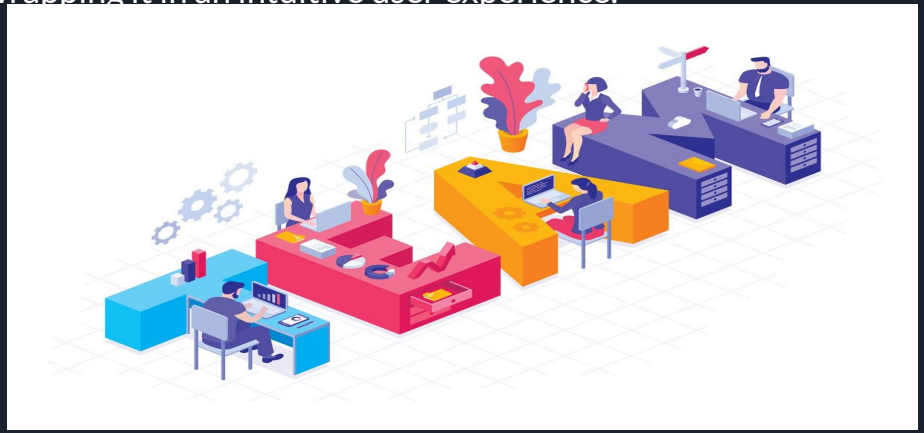
Testing and Infrastructure

Apollo performs a series of tests against each version, which are tracked, since they feel that testing is the most effective approach to guarantee that a product is efficient. To begin, the developers construct a test that is executed by the bots using their framework. For various experiments, the bot setups are updated on a regular basis. Secondly, when the developers add, delete, or update a test, they make an announcement so that everyone is aware of the changes. Finally, the findings are checked for any regressions.

Team Issues/Lessons Learned - 3

Core Principles

The goal of the developers is to maximize efficiency in order to make the platform speedier, which is accomplished by selecting appropriate engines and paying close attention to the pace of user interactions. Developers also prioritize security and stability, acting appropriately and applying traditional operating system design principles to the Apollo framework. Finally, despite the project making use of extremely advanced technology, there is an emphasis on simplicity by wrapping it in an intuitive user experience.



Conclusion

As a result of significant research and hard work, our group (FortyOne) believes that our conceptual architecture report clearly states and explains the architectural model and functionalities of Apollo. We have provided information on how the architecture, flow of data, concurrency, evolution of the system, and various use cases come together to provide the functionality that Apollo possesses.

In its five years of being on the market, Apollo has advanced significantly. Apollo's brain can be referred to as an "experienced AI driver", based on the AI system's ability to control the vehicle independent of a human driver. So far, the Apollo platform has logged over three million miles of road tests without any accidents and has carried over 100,000 passengers in 27 cities around the world.

This conceptual architecture model has enabled Apollo to become this successful in today's competitive world, that thrives on a positive user experience. Our group cannot wait to watch Apollo dominate, and become an autonomous vehicle powerhouse in the industry.

THANK YOU!