CISC 322

Assignment 1
Conceptual Architecture of Apollo

Sunday, February 20, 2022

FortyOne

Adam Cockell - 18aknc@queensu.ca
Ashton Thomas - 18ast8@queensu.ca
Dahyun JIN - 18dj8@queensu.ca
Kevin Subagaran - 19kks1@queensu.ca
Udbhav Balaji - 19ub@queensu.ca
Udit Kapoor - udit.kapoor@queensu.ca

Table of Contents

Abstract

Apollo is a flexible, open-source, comprehensive, and reliable software platform designed for partners in the automotive and autonomous-driving industries. This report will discuss in detail the various components and modules of the Apollo project to better understand the software style and architecture it uses.

The Apollo project has gradually evolved over the last decade. Our team explored and analyzed it based on the source code and previous versions of the system. It is important to note that there are extensive documents and information regarding the updates and architecture of Apollo. The scope of this report is limited to the open software platform, and information available to and analyzed by our team.

The architecture includes the following 5 platforms:

- Solutions
- Cloud Service Platform
- Open Software Platform
- Hardware Development Platform
- Open Vehicle Certification Platform

This report covers Open Software Platform which includes the following 8 modules:

- Map Engine
- Localization
- Perception
- Prediction
- Planning
- Control
- HMI
- Apollo CyberRT

Further details including concurrency, external interface, use cases and sequence diagrams will be discussed at length in this report.

Introduction and Overview

Apollo is an open-source, performance-focused framework designed for autonomous driving. Apollo enables safe, hands-free navigation of complex urban areas by implementing a robust platform which supports functionality from detecting surroundings to vehicle control systems. This is made possible using various sensors, including: LiDAR, radar, cameras, HDmap and other components interacting through inputs and outputs with algorithm-based processing in a pipe-and-filter architectural style. Additionally, the Apollo platform aims to be developer-friendly and flexible for a wide range of autonomous driving applications with minimal dependencies.

In runtime, the framework handles processing for the components using a scheduler which dynamically assigns tasks based on resources available, and the priority of each task. Tasks earlier in the execution pipeline are marked with higher priority, since they block processing in other components and introduce latency in the system. Proper scheduling allows the pipeline to run efficiently with minimal latency, facilitating the high performance of the Apollo framework. Additionally, the parallel computing model allows for high concurrency and throughput, ensuring optimized processing during runtime.

Apollo provides documentation and source code on their website, including previous versions on their GitHub repository. This will be the basis of our analysis of the Apollo software framework in this report. The architecture is composed of 5 platforms which are solutions, cloud service platform, open software platform, hardware development platform and open vehicle certification platform. This report will focus on the software platform. The software platform can be further broken down into 8 modules: map engine, localization, perception, prediction, planning, control, HMI and Apollo CyberRT.

This report aims to analyze the conceptual architecture behind Apollo and discuss the modules comprising the open software platform. We will look at global flow and data control and explore concurrency with how the system has evolved between updates over the past 10 years. Furthermore, the external interface will be discussed by looking at output generated by the Apollo system through use cases and sequence diagrams in order to illustrate the interactions between components within the system. The report will conclude with a summary of key findings, lessons learned, and a proposal for future development avenues.

Architecture

Modules

Localization

This module is responsible for performing all the required localization services. This is done in two ways: The RTK (Real-Time Kinematic) method, which uses GPS and IMU (Inertial Measurement Unit) information and the Multi-sensor fusion method, which uses data from GPS, IMU and LiDAR. These are the inputs for the Localization module.

This module is used instead of the Global Navigation Satellite System (GNSS) which has an accuracy of about 10m, due to satellite orbit and clock errors. These localization methods can provide centimeter positioning accuracy.

Perception

The perception module is responsible for finding 3-D obstacle tracks, as well as detecting and recognising traffic light states. As input, it takes LiDAR data, radar data, image data, as well as other raw data such as velocity and angular velocity of the host vehicle to produce the 3-D obstacle path, with information regarding velocity of the obstacles. It also performs traffic light state detection.

This module is responsible for ensuring that all the obstacles in the car's path are accounted for, as well as the right information is given to the Control module so that the car can continue its journey with no problems.

Prediction

This module is responsible for studying and predicting the behavior of obstacles detected by the perception module. It receives obstacle data, as well as basic perception information like positions, headings, velocities, etc. It also takes in localization information from the Localization module, in order to get a more accurate reading of the position of the car.

Once it completes execution, this module generates predicted trajectories, with risk probabilities for any detected obstacles.

Planning

The Planning module is responsible for finding a collision-free and comfortable navigation path. It takes in inputs from the Perception and Prediction modules, along with other raw data to calculate the optimal path to follow.

The module accounts for several important scenarios such as approaching traffic lights, parking, and emergency situations. It also needs to consider traffic rules to optimally plan paths.

Control

While the Planning module decides what trajectory to follow, it is the job of the Control module to physically control the vehicle. It uses different control algorithms to create a comfortable driving experience.

It issues commands to the steering wheel, throttle and brake to control the vehicle.

Global Flow and Control of Data

Data Flow

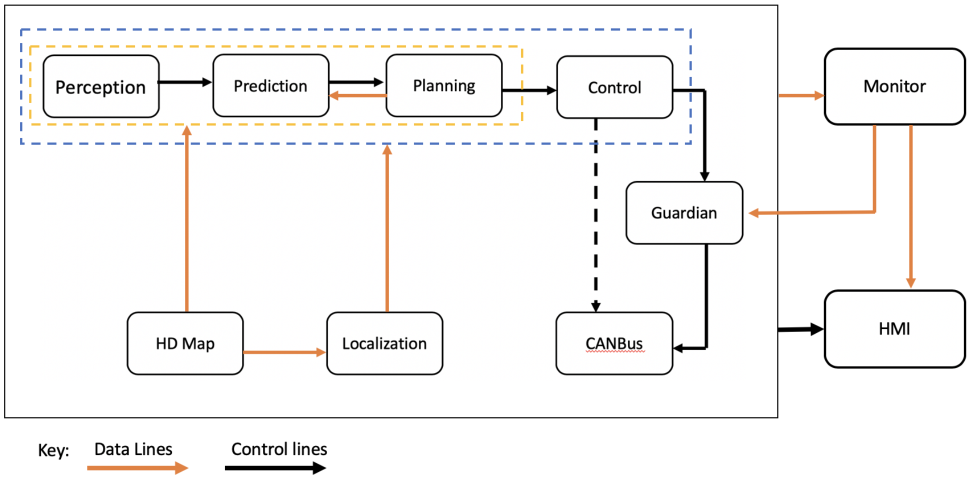
Data is collected from the HDMap module and is sent as input to the Localization, Perception and Prediction modules. Furthermore, the Localization module takes in data from the HDMap module and its output is sent as input to the Prediction and Planning modules. The Planning module also sends its data output to the Prediction module.

Once the Prediction module completes its computations, this data is sent to the Planning module so that appropriate trajectories can be computed based on the obstacles and their paths.

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 modules. The control is then handed over to the Guardian and CANBus modules. Next, the control is handed over to the HMI Module.

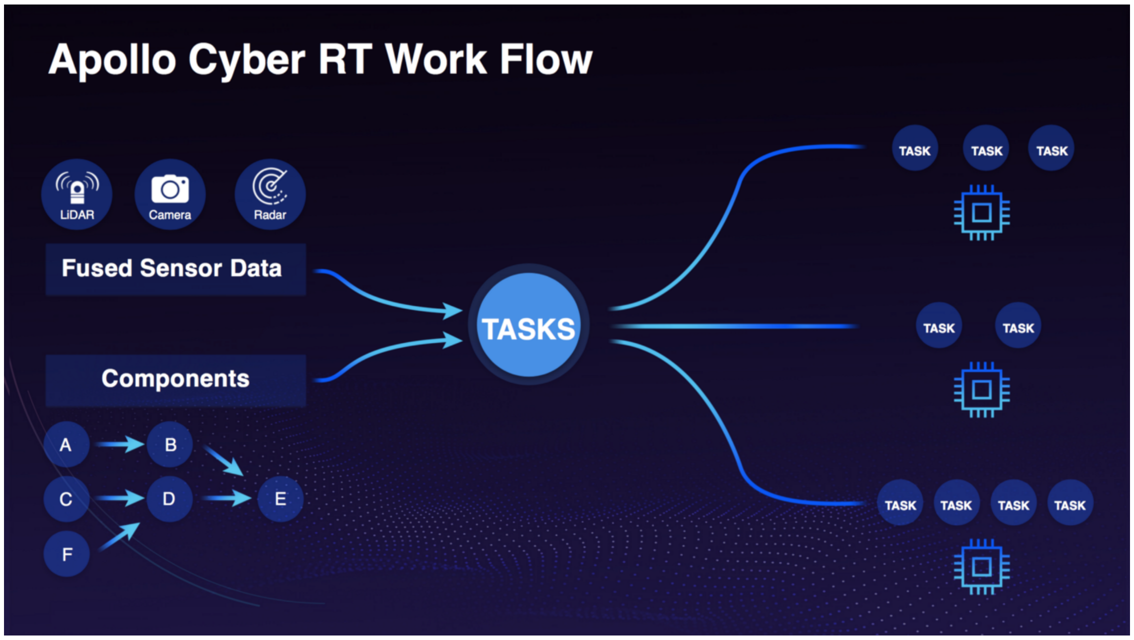
These flows can be understood better with the following diagram:



Concurrency

Autonomous driving is a computationally demanding task. It is necessary to not only have good hardware, but also optimized software to solve the complex tasks related to autonomous driving with as little latency between initial input and final output as possible.

To solve this issue, Apollo Cyber RT was developed as the world's first open-source runtime framework designed for development of autonomous driving technologies. The framework was first launched with Apollo 3.5.



The framework is built on top of the Apollo software modules. The components and their dependencies are linked together using a DAG dependency graph. The individual components are combined with raw sensor data to create lightweight jobs which are placed on a priority queue. The scheduler is resource aware, which means it knows what resources are available at a given time. The scheduler combines this information with the priority levels of the tasks to maximize throughput and lower latency.

A set of highly optimized threads are responsible for completing the tasks given by the scheduler. They work in parallel and enable high concurrency of task execution. All this ensures that the Cyber RT framework works at the highest level of performance without requiring heavy and complicated deployments.

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)

When it was first released, Apollo worked in an enclosed venue such as a test track or parking lot as the first version couldn't perceive obstacles in close proximity, drive on public roads, or drive in areas without GPS signals. Each platform except the Vehicle Platform got new features or improvements such as LiDAR, camera, black box, end-to-end security, and an updated HDMap as well as improvements to the Apollo runtime engine, perception and simulation. These improvements have enabled Apollo to test vehicles on simple urban roads autonomously.

2. Simple urban road -> Geo-fenced highways (v.2.0.0 - v2.5.0)

Cruising, avoiding collisions with leading vehicles and obstacle detection are very important for safety and when using the system in more general applications rather than using it in limited scenarios. As a result, Apollo focused on visualization and localization, and it led them to update the Open Software Platform (perception and planning) and Hardware Platform (camera) in this version.

3. Closed venue (v.2.5.0 - v3.0.0)

This version represents a major update to the architecture. A new module called Turnkey Solution is added for a total of 5 modules in the Apollo framework. This version focuses on allowing vehicles to drive in a closed venue setting at a low speed, so the new module includes dual car systems and low-speed driving in a closed venue with a minibus and micro-car, as well as valet parking. Furthermore, new modules and sensors are added, with the perception module update being particularly noteworthy in this case. A change was implemented to include camera data in visual localization, helping especially with differentiating between detected objects.

4. Complex driving scenarios (v.3.5.0 - v5.5.0)

The system is now able to provide 360-degree visibility and upgraded perception algorithms to handle changing conditions on road, making the system more secure and aware. During these updates, most components of the architecture are improved. Highlights in this architecture update include a new runtime framework called Apollo Cyber RT, V2X capabilities, and Open Vehicle Certification Platform.

5. Enhancing the Apollo Perception and Prediction modules (v.6.0.0 – v7.0.0)

Version 6.0 works with new data pipeline services to better serve Apollo developers. Based on this release, Apollo 7.0 incorporates 3 new deep learning models to enhance the capabilities for Apollo perception and prediction modules. Apollo studio is introduced in Cloud Service Platform utilizing the new data pipelines, and the PnC reinforcement learning model training and simulation evaluation service are published. This version update mainly focuses on the Cloud Service Platform and Open Software Platform.

External Interfaces

Dreamview

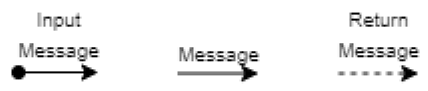
Dreamview provides a web application that can help developers better understand the outputs of other modules. It is a dynamic 3D HMI designed to help developers better understand the outputs of individual components of the system. It can display data from the Localization, Chassis, Planning, Monitor, 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.

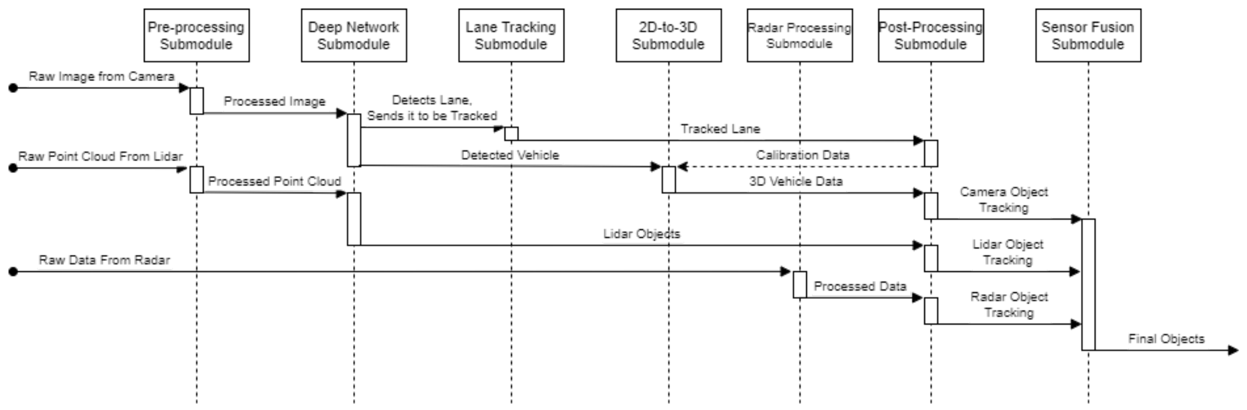
Protobuf Messages

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 Cases

Legend



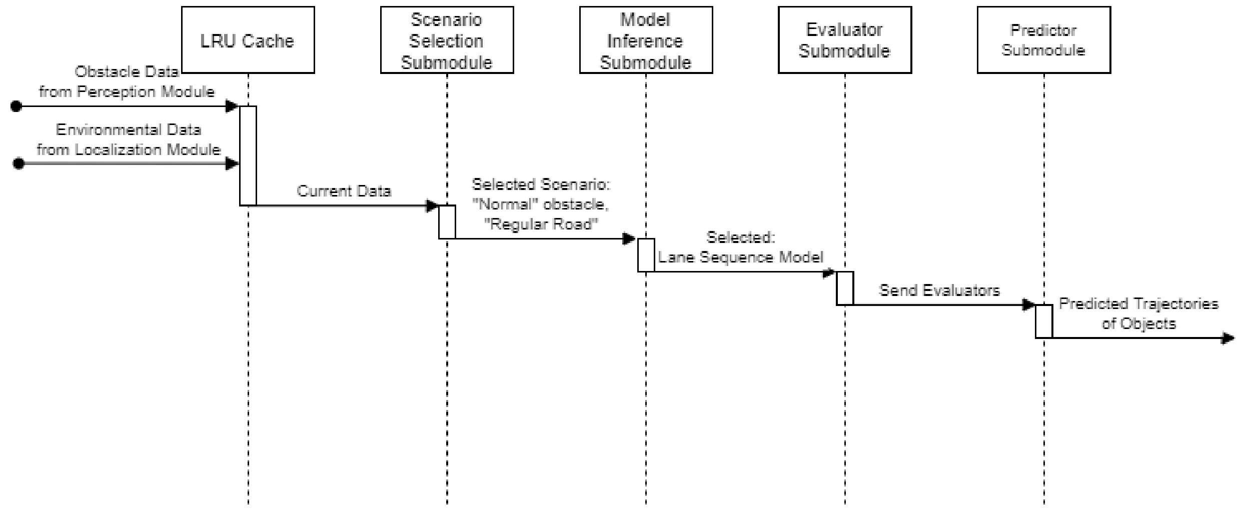


Use Case #1: Perception Module: A Vehicle Passes a Moving Apollo-Equipped Car

The Perception Module intakes data from an array of cameras, and radar and LiDAR sensors. The Perception Module first preprocesses the camera data and sends it to a Deep Network, where machine learning models identify and label the data captured from the image. The image from the camera is recognized as a vehicle. The 2D image of the vehicle is then converted to a 3D model, and then calibrated using environmental data extracted from the image.

The Perception Module then takes the raw output from the LiDAR sensors and preprocesses the point cloud data. The processed point cloud data is then sent to a Deep Network, where machine learning models label the data and identify 3D objects.

Next, the Perception Module takes the data from the radars and processes. Finally, the Perception Module combines the processed data from all three sensors and outputs obstacle data with positions, velocities, accelerations, and more.


LRU Cache
Scenario Selection Submodule
Model Inference Submodule
Evaluator Submodule
Predictor Submodule
Obstacle Data from Perception Module
Environmental Data from Localization Module
Current Data
Selected Scenario: "Normal" obstacle, "Regular Road"
Selected: Lane Sequence Model
Send Evaluators
Predicted Trajectories of Objects

Use Case #2: Prediction Module: A Vehicle Passes a Moving Apollo-Equipped Car

The Prediction Module receives input data from the Perception and Localization Modules. The data from the Perception Module contains information about obstacles. This information is added to the LRU Cache. The LRU Cache stores the most recently detected obstacles and environmental data, and is used by the Scenario Selection Submodule. The Scenario Selection Submodule runs continuously in the background and uses the detected obstacles from the LRU Cache and the environment information to select an appropriate scenario.

In the case of a vehicle merely passing by Apollo, it would determine the scenario to be a 'regular road' and the obstacle prioritization to be 'normal'. After the Scenario Selection Submodule has concluded it's process, it sends its results to the Model Inference Submodule which will provide an appropriate model for the current data to send to the Evaluator Submodule.

In this case, a Lane Sequence Model would be appropriate. The Evaluator Submodule will then choose multiple evaluators to predict the path and speed of the obstacle data. Finally, the Predictor Submodule predicts trajectories for the obstacles.

Data Dictionary

Architecture: High level structure and organization of the various subsystems of a software.

Parallel Computing: Using multiple processing elements simultaneously to solve a problem more efficiently.

Scheduler: The process responsible for assigning resources to perform tasks.

Thread: A thread is the smallest sequence of programmed instructions that can be managed directly by a scheduler.

Naming Conventions

DAG (Directed Acyclic Graph): A directed graph with no closed loops.

HMI (Human-Machine Interface): A User interface that connects a person to a machine or a system.

LiDAR (Light Detection and Ranging): Used for detecting distances to objects using a laser and measuring time before it bounces back to a receiver.

Team Issues/Lessons Learned

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. The business also announced the launch of the Apollo Fund, a $1.5 billion fund that would invest in 100 autonomous driving initiatives over the next three years.

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.

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.

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.

References

Apollo. (n.d.). Retrieved February 20, 2022, from https://apollo.auto/

ApolloAuto. (n.d.). *ApolloAuto/apollo: An open autonomous driving platform*. GitHub. Retrieved February 20, 2022, from https://github.com/ApolloAuto/apollo

Auto, A. (2019, February 1). *Apollo Cyber Rt-the runtime framework you've been waiting for*. Medium. Retrieved February 20, 2022, from https://medium.com/@apollo.baidu/apollo-cyber-rt-the-runtime-framework-youve-been-waiting-for-70cfed04eade

Baidu USA, LLC. (2017, September 21). *Baidu announces Apollo 1.5 and a 10 billion Yuan autonomous driving fund*. GlobeNewswire News Room. Retrieved February 20, 2022, from https://www.globenewswire.com/en/news-release/2017/09/21/1125651/0/en/Baidu-Announces-Apollo-1-5-and-a-10-Billion-yuan-Autonomous-Driving-Fund.html

Google. (n.d.). *Protocol buffers | google developers*. Google. Retrieved February 20, 2022, from https://developers.google.com/protocol-buffers

Luo, Q., Xu, K., Xiao, X., & Miao, J. (n.d.). *Data Driven Prediction Architecture for autonomous driving ...* Retrieved February 20, 2022, from https://www.researchgate.net/profile/Qi-Luo-2/publication/342169292_Data_Driven_Prediction_Architecture_for_Autonomous_Driving_and_its_Application_on_Apollo_Platform/links/5efba322a6fdcc4ca4406a9e/Data-Driven-Prediction-Architecture-for-Autonomous-Driving-and-its-Application-on-Apollo-Platform.pdf

Rapoza, K. (2020, September 16). *China's Baidu wants to be an autonomous vehicle powerhouse*. Forbes. Retrieved February 20, 2022, from https://www.forbes.com/sites/kenrapoza/2020/09/17/chinas-baidu-wants-to-be-an-autonomous-vehicle-powerhouse/?sh=3a15b1eb6307