The Conceptual Architecture of Apollo Platform

Authors:
Yuxuan Yang 18yy82@queensu.ca
Jialing Gong 18jg39@queensu.ca
Muyun Sai 18ms78@queensu.ca
Yuxin Huang 18yh98@queensu.ca
Hairuo Li 19hl12@queensu.ca
Songyu Yang 18sy36@queensu.ca

Abstract:
Apollo is a high performance, flexible architecture developed by Baidu, Kinglong and a consortium of more than 40 companies for the purpose of accelerating the development, testing, and deployment of Autonomous Vehicles.

This article first describes Apollo in terms of the functionality of the system and the interaction between its components, and then analyzes the conceptual architecture through system evolution, control and data flow, and concurrency, while illustrating the impact of the division of developer responsibilities on this.

This report provides new contributors and users of the system with architectural insight into the Apollo project, making it easier for them to understand or contribute to the project.

Introduction and Overview:

The report dive into the conceptual structure of the Apollo platform system. Specifically, in Functionality and Interaction part, it introduces how the architecture of a system refers to the decomposition of components and subcomponents which briefly means how each sensor in Hardware Development Platform is connected to the functionality Planning, Control, Perception in the Open Software Platform. Also, it discusses the potential capability of AVs (autonomous vehicles) to detect such as the obstacles or recognize the traffic condition by the reduction of human operation and the implementation of various sensors to incorporate the velocity, direction, and amounts of surrounding vehicles.

In The Evolvement for System part, the report introduces each version of Apollo platform start from 2007. It describes the key features and updates of each version. Moreover, it distinguishes the difference between coroutines and threads where threads are pre-emptively scheduled by the CPU, and coroutines are coordinated by user mode scheduling. It also introduces the core of Apollo's entire system which is the industrial computer IPC (Inter-Process Communication) and the function of this module Mobileye's RSS (Responsibility-Sensitive Safety) security system.

In Control and Data Flow part, it mainly describes how the Apollo system control the data and make them flowing among the parts. The automatic driving system first carries out the perception of the current environment (identification of vehicle and pedestrian traffic signs, etc.) and predicts the next development. Then the data is fed into the planning module to plan the subsequent orbit. After that, control module converts track data into control signals of vehicles and controls the vehicles through the car interaction module (CANBUS). At the same time, Guardian detects whether the program is wrong and finally displays it to users on the monitor.

In Concurrency part, it discusses how the main functions is used by the user (such as the Apollo module in the box in the figure) on the data transfer with the monitor and the control

of the HMI (A Human-Machine Interface). Also, it shows how the output of the Apollo software framework results in control commands that interact with the lower level module CANBus (Controller Area Network). Finally, the report depict how system process information and complete communication in an instant when then system involving self-driving.

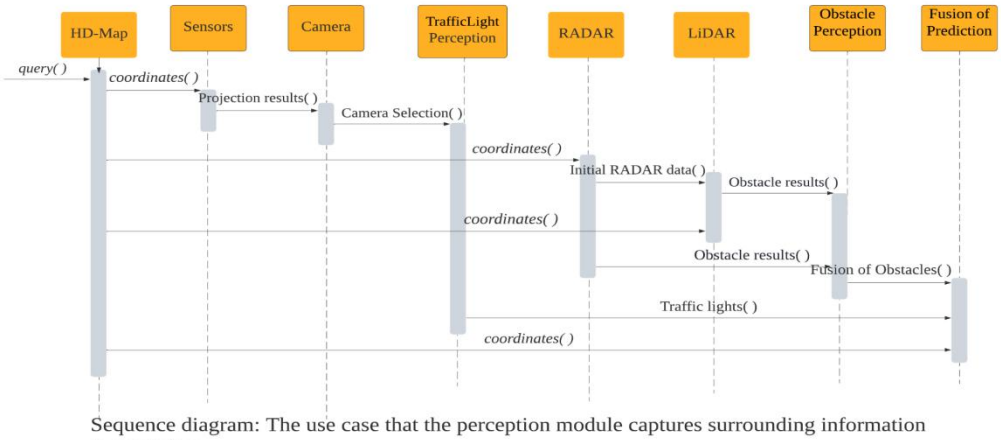
In Responsibility of Participating developers' part, the report describes the developers both from Apollo project in the company and also the developers from Open-source community. Both of them contribute to the whole Apollo open platform whereas the project developers are responsible for: designing various interfaces for data transmission; modifying the source code to different developers in the open-source community; keep updating or expanding the development environment. After different developers in the open platform receive the interface from Apollo, they can implement their own software or update the dataset.

Functionality and Interaction

The architecture of a system refers to the decomposition of components and subcomponents. Autonomous driving is mentioned as a complex system with several high-level functional components, such that perception, planning, and control of functions. [1]

The Apollo project contains the more concrete parts like localization, simulation, and HMI (Human Machine Interface) to strengthen performance of AVs (autonomous vehicles) when no human input. By the reduction of human operation, the potential capability of AVs is to detect the obstacles and recognize the traffic condition. [2] It implements various sensors to incorporate the velocity, direction, and amounts of surrounding vehicles. The perception plays a big role in detecting obstacles. The essential perceptions: obstacle perception and traffic light perception are divided into two or one sub-component. Its submodules LiDAR-based perception, RADAR-based obstacle perception, and HD-Map (High-Definition Map) respectively.[3]

Furthermore, obstacle perception based on LiDAR is based on full convolution depth neural network, which can predict the characteristics of obstacles, such as foreground probability and offset displacement. Then the object segmentation is realized based on these attributes. Radar based obstacle perception aims to process initial radar data. Usually, it will expand the track ID, eliminate noise, generate obstacle results, and filter the results according to ROI (region of interest). Obstacle result fusion is used to fuse LiDAR and radar obstacle results. On the other hand, the traffic light module obtains the coordinates of the traffic lights in front by querying the HD map. Using the internal and external parameters of the sensor, the traffic signal is projected from the world coordinates to the image coordinates. Then, camera selection is performed according to the projection results. Then use the surrounding bounding box to detect the traffic lights in ROI and recognize them as different color states.[3]



Sequence diagram: The use case that the perception module captures surrounding information to prediction.

Planning evaluates behavioral trajectories and motion logic of autonomous vehicles based on the traffic condition corresponding to collection of data. It is used for guiding a way which is more convenient and straightforward to reach a destination. The sub-components include mission planner, behavior planner, and local planner.[4]

To discuss the functionality of control, the modules monitor the motion of the vehicle with capability of braking, changing direction and improving traction control to handle some of the different road conditions. It can maximize vehicle traction to increase vehicle stability and provide more reaction time for steering, thereby preventing accidents. [5] The sub-components include anti-lock braking system (ABS), active front steering system (AFTS), and traction control. [1, 5]

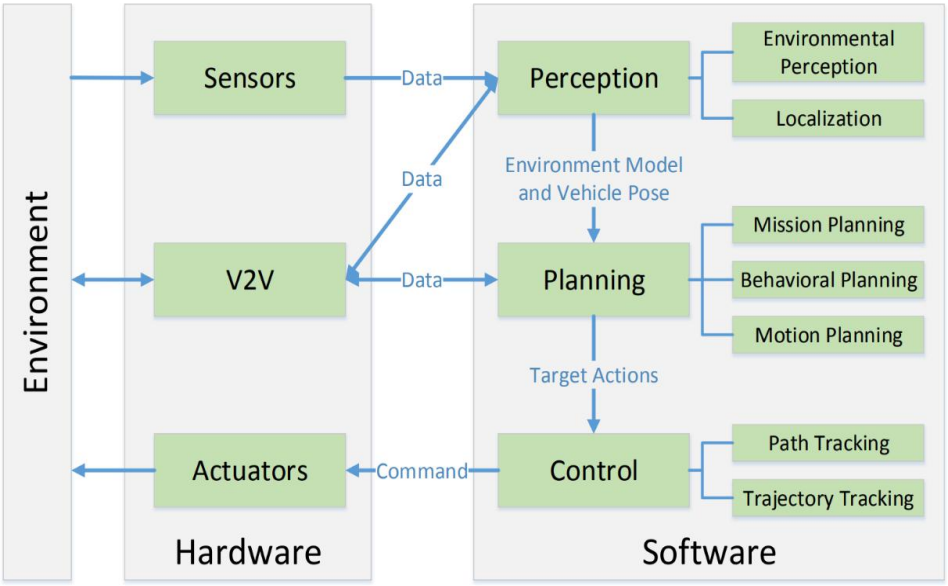
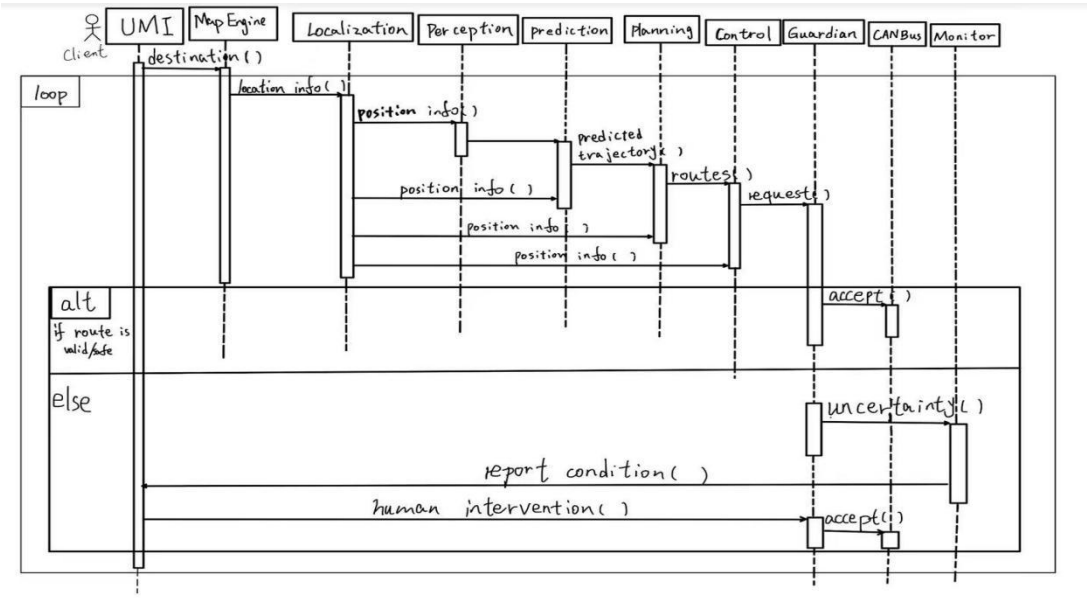


Figure1: Functionality and interaction between components and subcomponents in the software platform.

The hardware transfers data from sensors to perception systems. Each sub-component is interactive, completes its own task in the sub-part and returns one result to the next component. Both LiDAR-based perception and RADAR-based obstacle perception belong to environmental perception (shown in Figure1). LiDAR-based perception uses light detection to predict properties of obstacles. RADAR-based obstacle perception processes the initial RADAR and filters the result. [3, 8] They detect the information of surrounding objects, then cooperate with localization to send the environment model and vehicle pose to the planning component. Localization refers to a HD-map that accurately displays roads, directions and positions. It is part of perception to provide coordinates of traffic lights. [6].

The planning part has three sub-components. According to receiving data from perception, the mission planner makes decisions about which trajectory is optimal without consideration of static or dynamic objects. The top priorities point to constant traffic discipline, architecture distribution, or destination.[6] The behavior planner is responsible for instructing how vehicles vary based on unpredictable factors. [7] The local planner gathers



Sequence diagram: The use case that the client get to his/her destination by the autonomous vehicles.

the current information coming from behavior planner and mission planner. Then a suitable path is planned to reach the destination. [4, 6]

Finally, control receives the overall designed trajectory and executes action (shown in Sequence Diagram). Besides, control has the capabilities to detect the trail of a vehicle whether fit in design route, ABS and AFTS among which it is able to adopt measures in accidents due to operation error.

The Evolvement for System

There are 10 versions for the Apollo Autonomous Driving System which is apollo-v1.0.0, apollo-v1.5.0, apollo-v2.0.0, apollo-v2.5.0, apollo-v3.0.0, apollo-v3.5.0, apollo-v5.0.0, apollo-v5.5.0, apollo-v6.0.0, apollo-v7.0.0. For apollo-v1.0.0, this is the most basic version. It can only work in enclosed spaces (like the parking lot) and cannot sense obstacles at close range. Also, it cannot drive on public roads, and it cannot drive in areas without GPS(Global

Position System) signals. The version apollo-v7.0.0 is the most advanced and the smartest. It can automatically detect the surrounding environment and avoid obstacles in time.

In the version apollo-v3.5.0, there is a big update in which developers use their own CyberRT middleware for apollo-v3.5.0 and later versions. Here are some advantages of using CyberRT. There is no scheduling and non algorithmic operation logic. To solve this problem, the core design of CyberRT moves scheduling and tasks from the kernel space to the user space. Scheduling can be tightly integrated with algorithmic business logic. The reason for this is mainly the use of "coroutines". Threads are divided into kernel-mode threads and user-mode threads. User-mode threads need to be bound to kernel-mode threads. The CPU cannot perceive the existence of user-mode threads. It only knows that it is running one thread, which is actually a kernel-mode thread. User-mode threads have a name called coroutines. In order to facilitate the distinction, user-mode threads refer to coroutines and are used to refer to kernel-mode threads. There is a difference between coroutines and threads. Threads are preemptively scheduled by the CPU, and coroutines are coordinated by user mode scheduling. After a coroutine gives up the CPU, the next coroutine is executed. In addition, compared to Ros, CyberRT adds Component components, and components communicate through Cyber channel. In CyberRT, Message is used to implement inter-module communication, and its implementation is based on protobuf. At the same time, CyberRT also supports asynchronous computing tasks, optimizes thread usage and system resource allocation, and supports configuration files that define module topology. Of course, using CyberRT will also have disadvantages. CyberRT also has shortcomings with less user experience, and the resources are not as comprehensive as Ros(Robot Operating System).

The core of Apollo's entire system is the industrial computer IPC(Inter-Process Communication). In this configuration of Apollo, Neousys 6108GC is used, which is matched with Nvidia's GTX1080. The IPC receives the data sent by various sensors through the USB and Ethernet interfaces, and after processing, it connects to the CAN card(Controller Area Network) through the IPC(Inter-Process Communication) interface to finally drive the vehicle action. For the sensor array, in addition to the millimeter-wave radar, Lidar, Camera and ultrasonic radar all send the acquired information and data to the IPC(Inter-Process Communication) for processing through the USB interface and perform large-scale path planning and specific obstacle avoidance action planning. The output data of the millimeter-wave radar is not sent to the IPC(Inter-Process Communication) for processing but is directly provided to the CAN card(Controller Area Network) to connect to the CAN (Controller Area Network) bus of the vehicle. This whole process is a closed loop

The navigation module (IMU(Inertial Measurement Unit,)+GPS(Global Position System,)) uses the local HD Map(High-Definition Map) to perform the positioning operation and generates the position data of the relative map together with the output data of the perception module (scanning result of LiDAR). On the one hand, this position data can be directly provided to the path planning module (planning) for decision-making, and on the other hand, it can also control the "prediction module" (prediction) to predict the possible action trend changes near various moving and static targets. This trend prediction also ultimately affects the working status and results of the path planning module. Finally, the output of the Planning module includes not only the large path planning to reach the destination, but also the small obstacle avoidance, acceleration and deceleration, and overtaking action planning, and outputs the accelerator, direction, and brake of the CAN(Controller Area Network) Bus-driven vehicle.

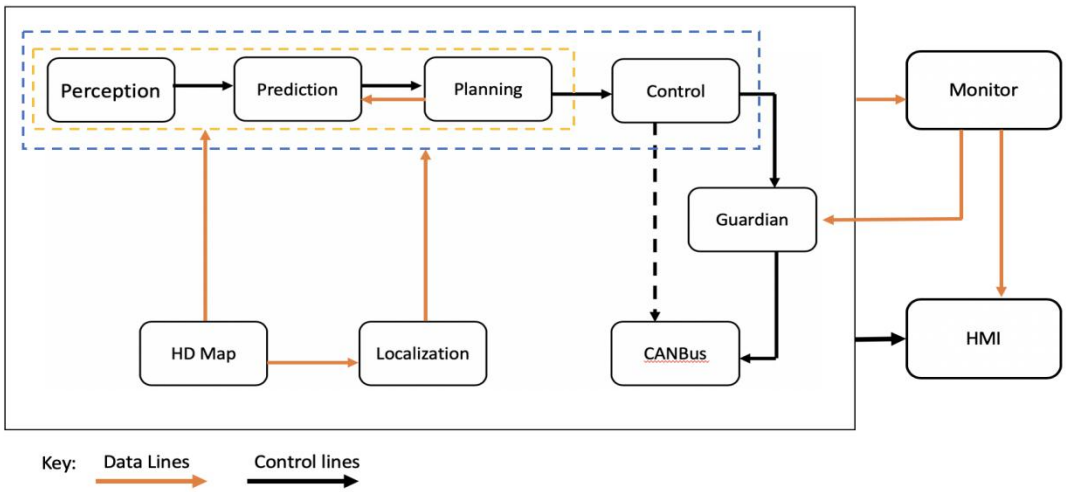
It is worth noting that the Guardian module is added between the above links. The function of this module is like Mobileye's RSS(Responsibility-Sensitive Safety) security system. Before the final vehicle performs an action, a discrimination mechanism is added to force the vehicle to meet certain safety criteria. The functions of Monitor and HMI(Human Machine Interface) are mainly displayed to the human driver on the HMI(Human Machine Interface) through the alarm mechanism of the Monitor when the vehicle cannot handle it or is determined by the system that humans must intervene.

Although self-driving cars are increasingly common in daily lives, the threats, and hidden dangers they pose cannot be ignored. On the evening of March 18, 2018, an Uber vehicle during a self-driving test crashed into a pedestrian in Tempe, Arizona. The pedestrian was taken to the hospital and later pronounced dead. According to preliminary findings, the Uber vehicle was in self-driving mode when it struck the pedestrian, the first case in history of an autonomous vehicle crashing and killing a pedestrian on a public road. This incident sounded the alarm bell about safety for unmanned driving. [8] From the current point of view, the current automatic driving system is not perfect and there are many safety hazards, such as the brakes cannot intervene, the radar cannot detect close-range objects, and so on. Therefore, researchers need to continue to improve in the future.

Control And Data Flow

In general situation control and data flow respectively is, the order in which individual statements, instruction or function calls of an imperative program are executed or evaluated and a broad concept, which has various meanings depending on the application and context.

Figure 1.Control and data flowGitHub - ApolloAuto/apollo: An open autonomous driving platform



The control and data flow of the software of the Baidu Apollo Auto can be divided into four levels. It is divided into two big parts which are software of the external information input and output device and the whole car control part. The car control is divided in the route planning part which used to plan the route, and the car control part which used to control the movement of the car.

The concepts of planning routes are the perception module, prediction module and planning module. The Baidu Apollo Auto has many cameras, radars and LiDARs(Laser

Radar) which can get the information from the obstacles and that will construct a HD Map of the surroundings. And the perception module function is to take the information of the obstacles surrounding the car then call the prediction module which is able to study and predict these obstacles, like car, roadblock and pedestrian, behavior with the information obtained from the perception module. After that the control module, which is the relatively complex part, will make the decision on how the car moves. It needs to make the car movement feasible but also needs to make the car movement smooth. Because the Baidu Apollo Auto is a commercial car, the route is feasible and passenger comfort is more important. The prediction module will control the planning module to make the best decision the planning module thinks, then the planning module will give back the decision it makes to the prediction module which will check the feasibility and use that to do the next step prediction.

Then the car movement gets planned, and now how to implement it. This part just adds the control module to the route planning part. The control module gets the information from the localization module, which gives the information about the car itself, then this module will change the orbital data to car control data which can be "understood" by the CANBus(Control Area Network Bus) module. Then it will use a guardian module to control the CANBus module to make the car make the movement that the planning module plans. The guardian module will be discussed with the monitor together.

Finally, the external information input and output device, the HMI(Human Machine Interface) module and the monitor module. The monitor module gets information from all the modules from above, to check if some problem in hardware or software exists, which means that they all run normally. If any part malfunctions the monitor will send the warning to the guardian module which will stop the car. In other situations, if the car works normally, the guardian will send all messages taken from the control module to the CANBus module just like it does not exist. Then the HMI module will output the information from all the modules mentioned before, also shows the warning, which gets from the Monitor module, if it exists.

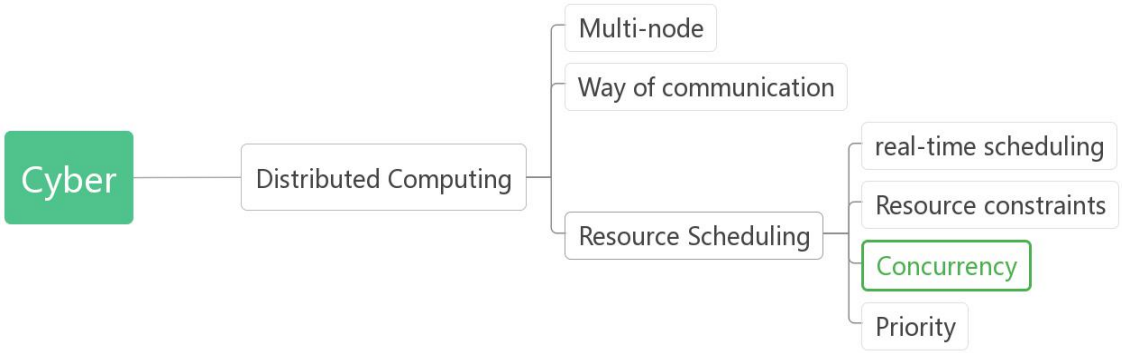
For example, when the car meets a traffic light and it is red. The perception module will find the distance to this cross and the color of traffic light. After collecting the information, prediction will be used to decide if the car can go through or not according to the information from the localization module like the speed, and car gear. Followed by planning, it will choose to stop the car and plan how to slow the car smoothly and feasible then stop. Then the planning module continues to give back the plan that it made to the prediction module. The control module will change the plan, which converts slowing then stopping the car to the language the car can execute. The vehicles tend to brake smoothly and stop before the car stop-line. After that, these instructions through the guardian module will check if there exist any problem. Then the CANBus module will get the instruction, then execute it. In the whole process, the monitor module works all the time to ensure process going well. Finally, the HMI will show to the passenger the final result.[9]

Concurrency

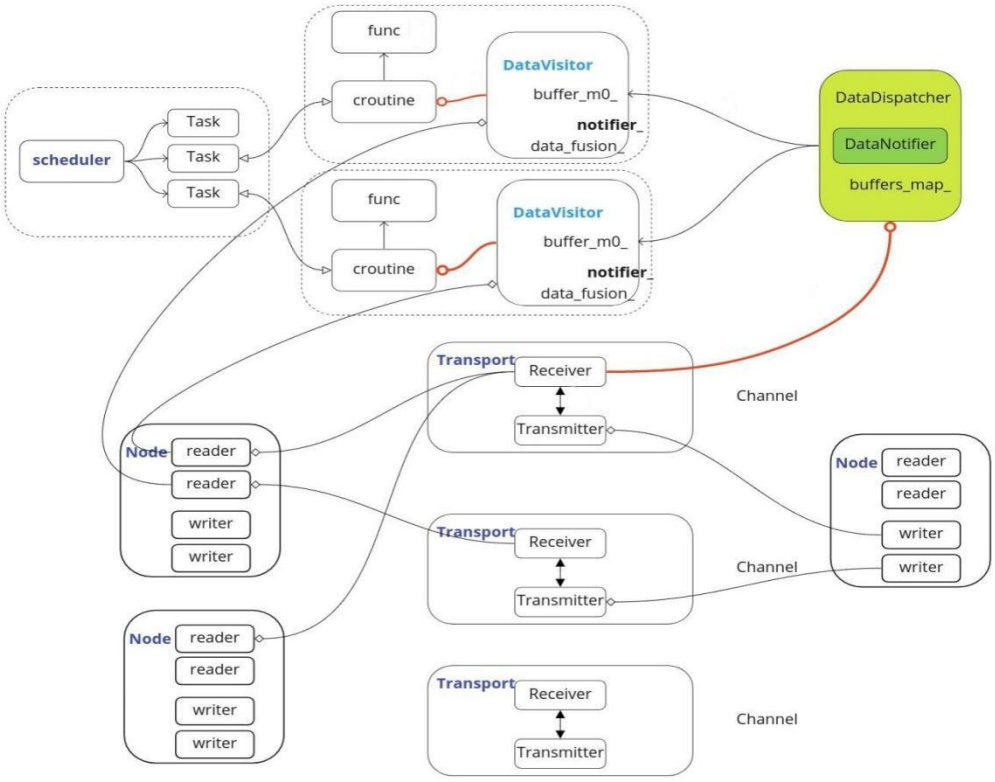
As all parts discussed above, the main functions used by the user (such as the Apollo module in the box in the figure) depend on the data transfer with the monitor and the control of the HMI(A Human-Machine Interface). the output of the Apollo software framework results in control commands that interact with the lower level module CANBus (Controller Area Network); at the same time the control module gets feedback signals from the underlying vehicle (the vehicle itself in vehicle reference frame: speed information, four-wheel speed information, vehicle health information, whether the chassis error message

information, danger information). Each module carries a part of independent functions and has certain data dependencies on each other. They actually run on a real-time operating system, which can be seen as the underlying framework of an operating system plus a message distribution mechanism. It requires the system to execute multiple processes concurrently, such as 1. Some robots carry multiple computers, each used to control part of the robot's drives or sensors. 2. even when there is only one computer, it is common to divide the program into smaller modules that run independently and collaborate with each other to accomplish complex control tasks 3. When multiple robots need to work together to complete a task, they often need to communicate with each other to support the completion of the task. [10]

Apollo v7 currently uses the Cyber system and can be the initial implementation of these needs.



It is important to be able to achieve such concurrency. Within the Apollo v7 system, multiple tasks are often generated simultaneously based on the current situation, perhaps from driving emergencies, dynamic changes in the external environment, or user command operations, etc., and different modules need to be mobilized. Since the system serves driving and involves the timeliness of judgments and user safety, tasks need to process information and complete communication in an instant. Compared to single-process processing, concurrent computing increases program throughput and has high input/output responsiveness[11], thus greatly enhancing NFR (non-functional requirements) while realizing the FR (functional requirements) of the architectural architecture. More specifically, say, midway through the use of Apollo autopilot, a car is driving on a highway and the service station to which the upcoming off-ramp leads is suddenly closed due to an accident. The next tasks that the system needs to handle are: receiving information about the change on the networked map (service station closure), updating the information on the current local map, notifying the user of the change using multimedia functions, re-routing, controlling the car on the new route, and so on. Some of these tasks are sequential and some are simultaneous. By completing the process earlier, the smoothness and economy of driving can be ensured. A similar data flow can be represented in Cyber as the following diagram, where a representation of concurrent task processing is shown.



Responsibilities of Participating developers

A good software project is based on good collaboration between each role in a developing team. Most of the time, when discussing participating developers, it is easier to connect them with the technical programmers who are responsible for the fundamental coding in the back end. Indeed, they are vital in the entire development. Furthermore, there are also many other roles like Project Manager; Architecture Analyst; System Analyst; Data-Base Administrator make a huge contribution to the whole system. One person may cover two or more roles. One role may be split between two or more people. In this case, close communication and collaboration between individuals become essential [13]. Thus, the division of responsibilities among participating developers is just like the root of a tree which is fundamental and important.

Specifically, in Apollo Open autonomous driving platform, it is comprised of four platforms: a Cloud Service Platform, an Open Software Platform, a Hardware Development Platform, and an Open Vehicle Certificate Platform. With the property of an open platform, not only do the developers from the Apollo project contribute to each platform, but there is also a large number of support from the open-source community. “Since Baidu’s autonomous driving platform Apollo was open-sourced in April 2017, it has built a community with over 45,000 developers, 210 industry partners, and compiled more than 600,000 source code lines in 97 countries around the world” [14]. This helps the platforms with system maintenance and updates for a better version. Most importantly, the contributors around the world take part in developing more self-learning functionalities in this huge deep-learning network with shared data and information. Apollo platforms also provide the developers of participating businesses and organizations with an open platform to develop their own services and devices.

Therefore, it is straightforward when discussing the responsibilities of each division among developers. In Apollo platforms, the developers are responsible for designing various

interfaces for data transmission and modifying the source code to different developers in the open-source community. Meanwhile, they also should keep updating or expanding the development environment. After different developers in the open platform receive the interface from Apollo, they can implement their own software or update the data set. For example, the perception part is one of the key parts in self-driving cars based on Apollo architecture. The self-driving cars should identify different perception tasks such as classification, detection, segmentation, and learning CNN (Convolutional Neural Networks) which are critical to perception [12]. In this case, developers in the open-source community can update various picture data for training to the cloud platform in the huge deep-learning network. It should also be possible for developers to develop their own Apollo software based on the shared data and source code.

Conclusions:

The main purpose of the Apollo autopilot system is to achieve L5 or even higher levels of autopilot, so that people only use cars instead of driving. The current Apollo autopilot system has been able to rely on its powerful computing and processing capabilities, and a variety of radars and cameras throughout the body of car to complete most of the autopilot functions. But it is not perfect and not reliable enough (problems like brake or detection failure are possible), so more developers and researchers are needed to continuously study, update and maintain its functions. All in all, with its large number of features, variability, and outstanding performance, it is an excellent system for autonomous driving goals right now. In the future, with the continuous improvement of the Apollo system, it will bring a more convenient and fast life to public.

Naming Conventions and Data Dictionary:

CANBus: Controller Area Net-work Bus, allow microcontrollers and devices to communicate with each other's applications without a host computer.

HMI(Human Machine Interface): “Human-machine interface”, also called human-machine, is a medium for interaction and information exchange.

LiDARs(Laser Radar): is a method for determining ranges(variable distance) by targeting and object with a laser

ROI (Region of Interest): In machine vision and image processing, the region to be processed is outlined from the processed image in the form of box, circle, ellipse, irregular polygon and so on.

ABS(Anti-lock Braking System): To restore traction to tires when vehicles steer in emergencies.

AFTS(Active Front Steering System): A steering system in which the relationship between the driver’s steer inputs and the angle of the steered road wheels may be continuously and intelligently altered.

CyberRT: The runtime framework.

Neousys 6108GC: The in-vehicle GPU computing edge AI platform.

Nvidia's GTX1080: The graphics cards.

IPC (Inter-Process Communication): Inter-Process Communication refers to the interaction between the data of two processes.

CAN card (Controller Area Network card): It is used to collect information of each node on the CAN bus.

Ros (Robot Operating System): ROS is a highly flexible software architecture for programming robot software.

IMU (Inertial Measurement Unit): IMU can be integrated into GPS based automotive navigation systems or vehicle tracking systems.

GPS (Global Position System): It is the satellite-based radionavigation system owned by United States government.

HD Map (High-Definition Map): It is a highly accurate map used in autonomous driving.

Mobileye's RSS security system (Mobileye's responsibility-sensitive safety security system): It is a mathematical model for automated vehicle safety.

CNN (Convolutional Neural Networks): a class of artificial neural network, most commonly applied to analyze visual imagery.

References

[1] Qiaochu Guo(April 29, 2020). Software System of Autonomous Vehicles: Architecture, Network and OS.

[2] P. Penmetsa; E. Kofi Adanu; D. Wood; T. Wang; S. L.Jones(June 2019). doi: https://doi.org/10.1016/j.techfore.2019.02.010

[3] https://apollo.auto/platform/perception.html

[4] Dhanoop Karunakaran(Nov 25, 2020). Motion planning module in Autonomous Vehicle — Mission planner. doi: https://medium.com/intro-to-artificial-intelligence/motion-planning-module-in-autonomous-vehicle-mission-planner-671b2155dec1

[5] P. Falcone; F. Borrelli; J. Asgari; H. Eric Tseng; D. Hrovat; Fellow (MAY 2007). Predictive Active Steering Control for Autonomous Vehicle Systems.

[6] S. Drew Pendleton; H. Andersen; Xinxin Du; Xiaotong Shen; M. Meghjani; You Hong Eng; D. Rus; M. H. Ang Jr (17 February 2017). Perception, Planning, Control, and Coordination for Autonomous Vehicles.

[7] Maggie Weeks(May 28, 2019). Autonomous Driving and the Need For Motion Planning. doi: https://rtr.ai/autonomous-driving-and-the-need-for-motion-planning/

[8] https://blog.csdn.net/qq_45577461/article/details/107451042

[9] GitHub - ApolloAuto/apollo: An open autonomous driving platform

[10] https://zhuanlan.zhihu.com/p/62259081

[11] https://en.wikipedia.org/wiki/Concurrent_computing

[12] https://apollo.auto/devcenter/devcenter.html

[13] Eric Walz(May 27, 2020). China's Baidu Completes its ‘Apollo Park’, the World's Largest Autonomous Driving & Intelligent Vehicle Testing Site. doi: https://www.futurecar.com/3946/Chinas-Baidu-Completes-its-Apollo-Park-the-Worlds-Largest-Autonomous-Driving-&-Intelligent-Vehicle-Testing-Site

[14]Apollo Auto (Oct 10, 2020). Inside Apollo 6.0: A Road Towards Fully Driverless Technology. doi: https://medium.com/apollo-auto/inside-apollo-6-0-a-road-towards-fully-driverless-technology-522b2b4295cc