Analysis of execution delay of Autoware Nodes

Created by carmark chen
Last updated: Jul 11, 2022 by Shark Liu • 3 min read • • 5 people viewed

Introduction

The purpose of this work is to find out the quantitative performance data of the new Autoware by analyzing the excution delay of each function nodes.

The excution delay is recorded and analyzed using the following tools:

- ros2_tracing
- tracecompass

The test setup is based on amd64 PC and real sensors. The hardware platform and main sensors we use are as follows:

Platform	ADLINK MVP-6100 • CPU: i7-9700E • RAM: 32G • GPU: Quadro P2200
System	Ubuntu 20.04 ROS2 Galactic
Lidar	Ouster OS1-64
Camera	FLIR Firefly • 1440*1080, 30 FPS

Execution delay data

The setting of test procedure:

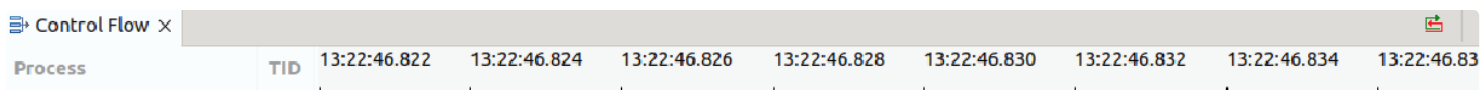
1. Re-write the launch files for all main modules so that each node is launched outside ros containers, so that the excution time of each node can be recorded, otherwise, the nodes in one container could only be recorded as a single process.
2. Start the ros2_tracing tool and then run AW, keep logging for a period of time, e.g. 5-10s, and save the trace.
3. Use tracecompass to proceed the trace, measure the execution time of each node for 10 times, to calculate the average, minimum and maximum execution time.

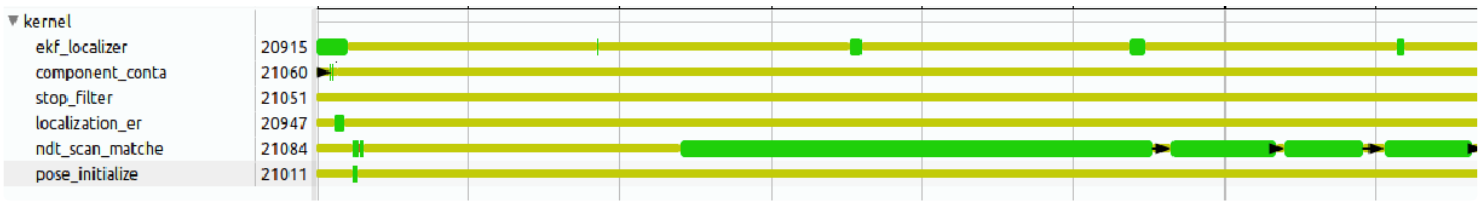
The execution delay of main modules such as localization, perception, planning and control and their sub modules are recoreded and analized as follows:

- **Localization**

In the localization module, we analyzed the running time of nodes such as pose_initializer, ndt_scan_matche, ekf_localizer, and the container which executing the pointcloud preprocessor. Pose_initializer which costs about 6ms only works at startup of localization, and is not included in the total time analysis.

1. Trace





2. Time

Sub modules	Sampling time										Avg (ms)	Min (ms)	Max (ms)
ekf_localizer	2.1	1.3	3	1.2	1.8	1.4	2.7	1.9	1	2.2	1.86	1	3
locali_error_monitor	0.25	0.28	0.26	0.456	0.11	0.12	0.207	0.322	0.517	0.385	0.29	0.11	0.517
ndt_scan_matcher	8.1	7.9	10.5	8.5	9.7	9.6	9	11.1	12.0	18.7	10.5	7.9	18.7
stop_filter	0.39	0.42	0.21	0.25	0.36	0.31	0.19	0.21	0.20	0.18	0.27	0.19	0.42
pose_initializer	2.0	3.1	6.3	7.6	3.8	2.2	6.2	2.4	18.8	8.1	6.05	2.0	18.8
component_contain	6.8	7.3	6.1	5.2	6.9	10.4	7.2	6.5	11.6	8.7	7.67	5.2	11.6

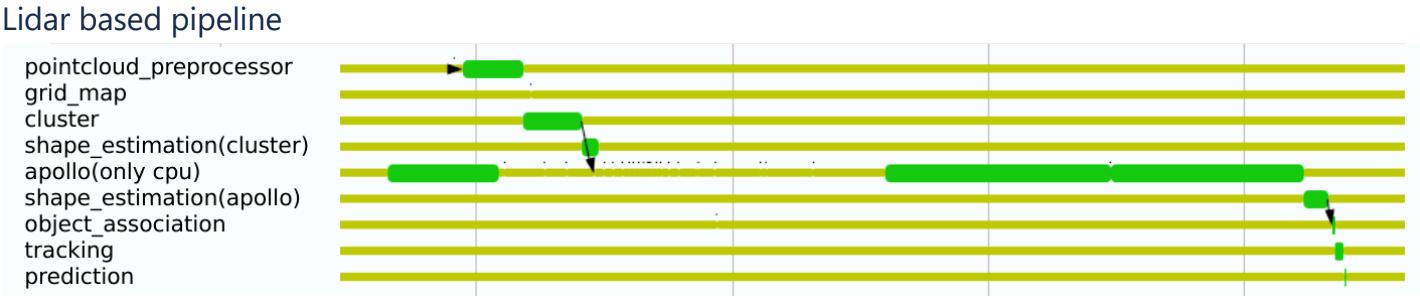
The total time consumed by the location module is about: 20ms

- **Perception**

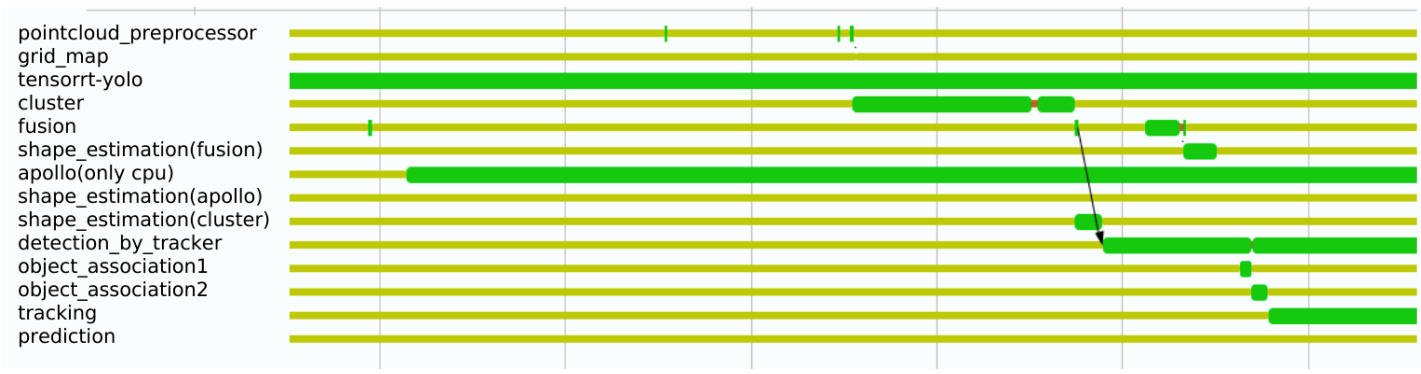
In the perception module, we analyzed the lidar based perception and lidar_camera perception. In detection modules such as yolo and apollo_cnn, GPU and CPU are both used, we only analyze the running time on CPU in this paper. Execution time is correlated with the number of obstacles in the environment. There are 2-4 obstacles in the statistical scene.

1. trace

a. Lidar based pipeline



b. Lidar/Camera fusion pipeline



2. time

Sub modules	Sampling time										Avg (ms)	Min (ms)	Max (ms)
Lidar based perception													
pointcloud_preprocessor	4.6	5.5	6.1	7.7	5.7	5.1	7.8	8.9	6.3	5.2	6.29	4.6	8.9
grid_map	0.005	0.005	0.005	0.003	0.004	0.004	0.004	0.005	0.005	0.005	0.0045	0.003	0.005
cluster	4.8	5	5.7	4.8	5.3	4.9	4.8	4.8	4.9	6	5.1	4.8	6
shape_estimate(cluster)	1.2	1.3	1.5	1.3	1.4	1.4	1.5	1.8	1.3	1.8	1.45	1.2	1.8

apollo(cpu)	47.7	50.4	48.9	47.1	47.7	46.4	45.6	40.3	47.8	45.3	46.72	40.3	50.4
shape_estimate(apollo)	2	2.1	2.1	2.2	2.1	2.1	2.1	2.4	1.9	2	2.1	1.9	2.4
object_association	0.27	0.24	0.21	0.26	0.25	0.25	0.23	0.22	0.38	0.23	0.254	0.21	0.38
tracking	0.51	0.56	0.55	0.55	0.6	0.64	0.83	0.63	0.64	0.62	0.613	0.51	0.83
prediction	0.05	0.052	0.053	0.054	0.05	0.049	0.072	0.05	0.081	0.049	0.056	0.05	0.08
Lidar_Camera perception													
pointcloud_preprocessor	4.8	7.2	5.6	3.5	4.9	5.1	5.4	4.8	4.9	6.1	5.23	3.5	7.2
grid_map	1.3	4.2	1.3	1.2	3.1	2.2	2.5	1.8	2.3	2	2.19	1.2	4.2
tensorrt-yolo(GPU)	\	\	\	\	\	\	\	\	\	\	\	\	\
cluster	5.4	5.2	5.6	5.6	5.0	5.6	3.8	5.5	6.1	5.8	5.36	3.8	6.1
fusion	0.85	0.94	1.10	0.92	0.81	1.02	0.84	0.99	1.05	0.98	0.95	0.81	1.1
shape_estimat(fusion)	1.58	0.87	0.75	0.76	0.81	0.8	0.95	0.97	0.87	1.13	0.949	0.75	1.58
apollo (only cpu)	45	48	45	46	49	48	47	49	44	47	46.8	44	49
shape_estimat(apollo)	1.5	1.6	1.6	1.5	1.5	1.6	1.5	1.4	1.5	1.7	1.54	1.4	1.7
shape_estimatcluster	2.3	2.2	2	1.9	2.1	2.1	2.3	2.2	2	2.2	2.13	1.9	2.3
detection_by_tracker	9.9	10.9	11.8	9.8	8.6	9.1	9.5	10.2	10.1	11.3	10.12	8.6	11.8
object_association1	0.27	0.24	0.28	0.26	0.25	0.25	0.25	0.27	0.25	0.24	0.256	0.24	0.28
object_association2	0.35	0.23	0.27	0.42	0.23	0.29	0.25	0.27	0.46	0.28	0.305	0.23	0.46
tracking	3	3.1	3.3	5.7	2.9	4.5	3.8	3.9	5.1	5.7	4.1	2.9	5.7
prediction	0.065	0.061	0.064	0.031	0.066	0.049	0.088	0.051	0.053	0.057	0.059	0.03	0.09

The total time of cpu consumed by the perception module are as follow:

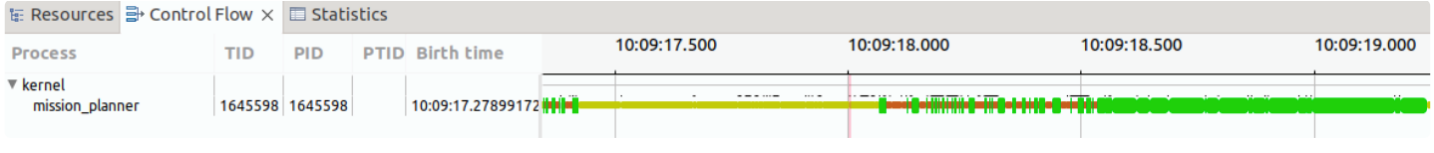
Lidar based perception: 62ms

Lidar_Camera perception: 78ms

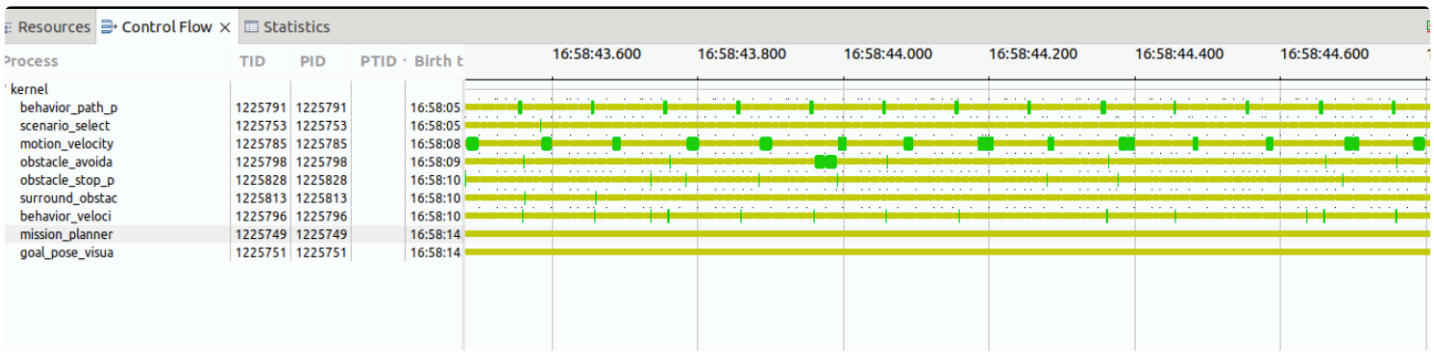
- **Planning**

In the planning moudule, the mission_planner which costs about 1 second to load the vector map and generate global path only works at startup of planning. So, it is not included in the total time analysis. In obstacle_avoidance_planner, replan needs more time than the normal.

1. trace
    a. mission_planning



b. main_planning



2. Time

Sub modules		Sampling time										Avg (ms)	Min (ms)	Max (ms)
mission_planner		995	1010	980	990	1015	1004	996	988	1011	1003	999	980	1015
behavior_path_planner		7.4	6.3	6.9	7.4	9.7	7.2	7.3	7.2	7.8	7.9	7.51	6.3	9.7
behavior_velo_planner		4	3.9	4.2	3.5	4.9	4.1	3.8	3.9	3.7	4	4	3.5	4.9
obstacle_avoidance_planner	replan	21	23.6	33	30	19.8	20	21.6	23.2	17.7	18.9	22.8	17.7	33
	normal	1.1	1.2	1.2	1.7	1.9	1.3	2	1.5	1.8	1.2	1.49	1.1	2
surround_obstacle_check		0.61	0.71	0.72	0.69	0.72	0.73	0.61	0.59	0.77	0.6	0.68	0.59	0.77
obstacle_stop_planner		1.9	1.5	1.3	1.7	1.3	1.2	1.8	1.3	1.2	1.9	1.51	1.2	1.9
scenario_selector		0.53	0.55	0.53	0.48	0.56	0.57	0.64	0.66	0.47	0.56	0.555	0.47	0.66
motion_velocity_smooth		6.7	7.3	9.5	12.4	13.9	9.9	10	11	10.3	11.1	10.2	6.7	13.9

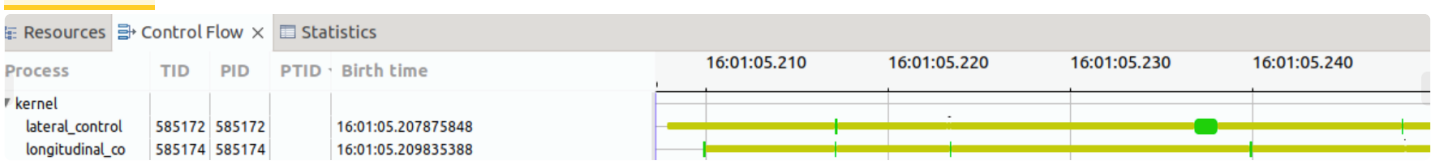
The total time consumed by the planning module is about: 50ms

- **Control**

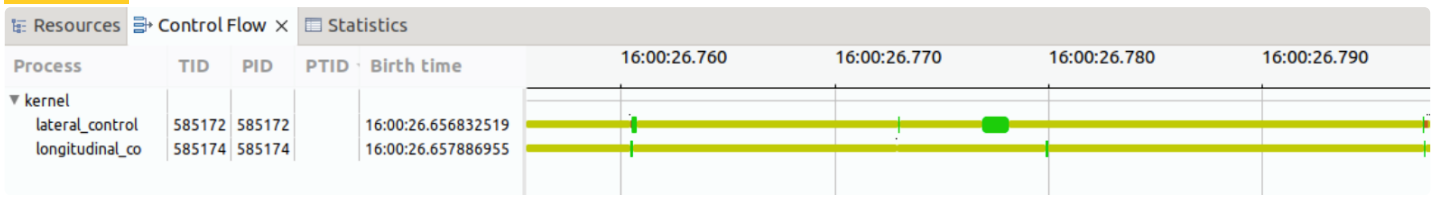
In control module, we focus on two controllers (MPC for lateral and PID for longitudinal), and analyze the execution time of the controllers when the car is in movement state and standstill state respectively.

1. Trace

a. movement



b. standstill



2. Time

Sub modules	Sampling time										Avg (ms)	Min (ms)	Max (ms)
Movement													
lat	1.2	1.7	1.8	2.2	2.5	1.7	2.4	1.3	2.2	2.0	1.9	1.2	2.5
lon	0.14	0.09	0.10	0.16	0.12	0.10	0.20	0.22	0.15	0.12	0.14	0.09	0.22
Standstill													
lat	1.6	3.8	3.8	1.9	1.2	2.7	2.2	2.5	2.9	3.3	2.59	1.6	3.8
lon	0.32	0.32	0.18	0.11	0.12	0.20	0.08	0.14	0.21	0.09	0.177	0.08	0.32

ICH													

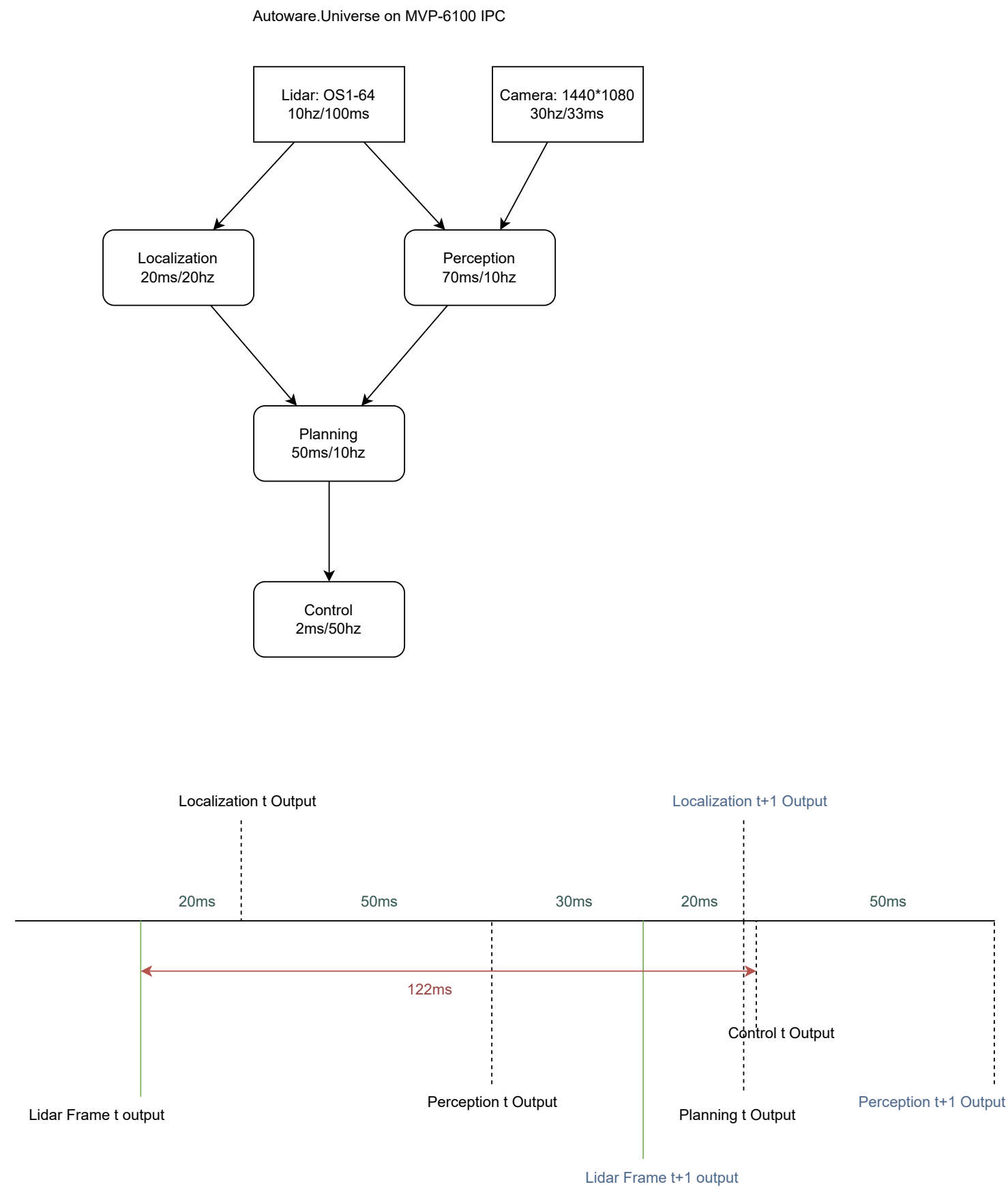
The total time consumed by the control module are as follows,and we can find execution time have no relation with the car speed.

Movement: 2.0ms

Standstill: 2.767ms

Summary

It can be concluded from the above data, the latency of Autoware in our reference system is shown in the follow figure.



Be the first to add a reaction

No labels