Worse performance compared to Architecture Proposal ROS1 version? #2636





VRichardJP on May 30, 2022 (Collaborator)

I have been using ROS1 version of

https://github.com/tier4/AutowareArchitectureProposal.iv in our vehicle for quite some time, and I have recently started to pick up with autoware.universe.

After a few tests on field, I can't help but feel that the overall performance of autoware on my machine is worse than what it used to be with AAP ROS 1. For instance, with AAP I did not have any localization issue on our test field, but with universe the localization is quite unstable and is easily lost if I drive too fast or turn suddenly. Similarly, the whole object tracking pipeline seems to be slower than with AAP (1~2 second delay from VLP cloud to tracked objects). So far, I have only been able to reach real-time performance by heavily downsampling the VLP sensor cloud (which was not necessary with AAP).

I don't doubt universe algorithms and features are better and more reliable than AAP, but I somewhat expected that the change from ROS1 to ROS2, the use of efficient DDS implementation and intraprocess communication would compensate the extra processing.

Is it only my experience/feeling? Is autoware heavier (slower?) than before or it just a matter of configuration/tuning? (e.g. #204 (comment))

1

Answered by VRichardJP on Jun 2, 2022

I think I finally managed to reach a good performance!

At the end of the day, it was only a matter of a few changes:

Tweaking BIOS and governor configuration to max out the CPU

View full answer ↓

O comments · 11 replies

Oldest

Newest

Top



kenji-miyake on May 30, 2022

Category

Q&A

Labels

None yet

3 participants





Although I'm not so familiar with this performance issue, I remember one of the biggest reasons is the overhead of existing executors.

I hope Events Executor, which is released in Humble, will improve the performance.

https://discourse.ros.org/t/ros2-middleware-change-proposal/15863

Regarding Autoware itself, I believe it didn't get so slower than TIER IV's proposal version. But as the features have been increasing gradually, it requires more machine resources if you use the full features.

1 2

1 reply



kenji-miyake on May 30, 2022

Also, if you want to optimize performance, you need to tune your DDS settings.

I asked my colleague @takam5f2 to provide some information about this.



takam5f2 on May 30, 2022 (Collaborator)

@VRichardJP

After migrating from ROS 1 to ROS 2, we tend to take care of performance issue as you mentioned. There are many possible cause to degrade performance; executors, multicast communication, and so on. As you know, unlike ROS 1, ROS 2 does not have master node, so that all nodes try to discover another to communicate autonomously. Consequently, ROS 2 requires more workload of nodes than ROS 1.

For instance, with AAP I did not have any localization issue on our test field, but with universe the localization is quite unstable and is easily lost if I drive too fast or turn suddenly. Similarly, the whole object tracking pipeline seems to be slower than with AAP (1~2 second delay from VLP cloud to tracked objects).

However, we have not confronted with such disastrous performance issue as you said. Considering behavior you mentioned, I suspects that your system has something wrong with DDS configuration or memory bandwidth.

but I somewhat expected that the change from ROS1 to ROS2, the use of efficient DDS implementation and intraprocess communication would compensate the extra processing.

Inter-process communication via multicast on DDS costs CPU time. Each ROS 2 process has a thread to receive topic message, named "recvMC". recvMC threads cost about 20-30% of total CPU time in Autoware. Please try unicast instead of multicast if you have not tried yet. In TIER IV, it is recommended to set spdp on AllowMulticast to use unicast mainly as below.

You will find some useful references on CycloneDDS.

CycloneDDS

Options

```
<?xml version="1.0" encoding="UTF-8" ?>
<CycloneDDS xmlns="https://cdds.io/config" xmlns:xsi="http://w.....3</pre>
    <Domain Id="any">
        <General>
            <Interfaces>
                <NetworkInterface autodetermine="true" priority="de-</pre>
            </Interfaces>
            <AllowMulticast>spdp</AllowMulticast>
            <MaxMessageSize>65500B
        </General>
        <Discovery>
            <EnableTopicDiscoveryEndpoints>true</EnableTopicDiscove</pre>
        </Discovery>
        <Internal>
            <Watermarks>
                <WhcHigh>500kB</WhcHigh>
            </Watermarks>
        </Internal>
        <Tracing>
            <Verbosity>config</Verbosity>
            <OutputFile>cdds.log.${CYCLONEDDS_PID}</OutputFile>
        </Tracing>
    </Domain>
</CycloneDDS>
```





3 replies



takam5f2 on May 30, 2022 (Collaborator)

I don't doubt universe algorithms and features are better and more reliable than AAP

We added new features to Autoware after migrating ROS 2. Compared with ROS-1 based Autoware implemented till last year, current version requires more computing resources than it.

Sorry, but I can't describe it quantitatively.



VRichardJP on May 31, 2022 (Collaborator) (Author)

Using custom cycloneDDS configuration seems indeed to help quite a lot.

The xml configuration you gave is for 0.9.x version I think, but it seems it is possible to get the equivalent for 0.8.0 (the one bundled with galactic)

```
</Watermarks>
        </Internal>
        <Tracing>
            <Verbosity>config</Verbosity>
            <OutputFile>/home/sig/.ros/cdds.log.${CYCLONEDDS_F
        </Tracing>
    </Domain>
</CycloneDDS>
```

Regarding cycloneDDS version, is there any significative advantage using newer version?



takam5f2 on May 31, 2022 (Collaborator)

The configuration looks good.

is there any significative advantage using newer version? I believed that it is improved day by day, but we will evaluate it during migration from Galactic to Humble.



VRichardJP on May 31, 2022 (Collaborator) (Author)

To illustrate my situation, I have made a small script to track message frequency (like ros2 topic hz) and age (msg.header.stamp - recv_time) over few key topics. Autoware performance is deeply impacted by the choice of DDS and its configuration.

In the following situation I use cycloneDDS with its default configuration:

•				_		
TOPIC NAME	- 1	HZ	- 1	AGE	-1	SINCE LAST
/sensing/lidar/top/velodyne_packets	i_	9.842	- 1	0.105	- Í	0.023
/sensing/lidar/top/pointcloud	- i	9.694	- 1	0.196	- 1	0.035
/sensing/lidar/concatenated/pointcloud	- 1	9.747	- 1	0.202	- 1	0.027
/localization/util/measurement_range/pointcloud	- 1	9.787	- 1	0.273	- 1	0.018
/localization/util/voxel_grid_downsample/pointcloud	- 1	2.737	- 1	0.645	- 1	0.273
/localization/util/downsample/pointcloud	- 1	2.738	- 1	0.647	- 1	0.271
/localization/pose_estimator/pose_with_covariance	- 1	2.741	- 1	0.661	- 1	0.243
/perception/object_recognition/detection/apollo/labeled_clusters	- 1	3.917	- 1	0.483	- 1	0.040
/perception/object_recognition/detection/apollo/objects	- 1	2.686	- 1	0.968	- 1	0.221
/perception/object_recognition/detection/apollo/validation/objects	- 1	0.646	- 1	2.118	- 1	0.965
/perception/object_recognition/detection/objects	- 1	0.687	- 1	3.517	- 1	0.597
/perception/object_recognition/tracking/objects	- 1	0.649	- 1	3.486	- 1	0.584
/perception/object_recognition/objects	i i	0.649	İ	3.491	İ	0.581

Topics are ordered. For example the VLP preprocessing pipeline from top/velodyne_packets to top/pointcloud takes 0.196 - 0.105 = 91ms. Another example: the VLP scan is already 661ms old by the time the new localization is computed

(/localization/pose_estimator/pose_with_covariance). It gets worse as we go down, with tracked objects being already 3 seconds old by the time there are published.

As I said, I need to heavily downsample the VLP cloud to get "acceptable" performance:

```
TOPIC NAME

/sensing/lidar/top/velodyne_packets

/sensing/lidar/top/pointcloud

/sensing/lidar/concatenated/pointcloud

/localization/util/measurement_range/pointcloud

/localization/util/voxel_grid_downsample/pointcloud

/localization/util/downsample/pointcloud

/localization/pose_estimator/pose_with_covariance

/perception/object_recognition/detection/apollo/labeled_clusters

/perception/object_recognition/detection/apollo/objects

/perception/object_recognition/detection/apollo/validation/objects

/perception/object_recognition/detection/objects

/perception/object_recognition/faction/objects

/perception/object_recognition/objects

/perception/object_recognition/objects
  OPIC NAME
                                                                                                                                                                                                                                                                                                                                                                                                                                                                              SINCE LAST
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       0.037
0.056
                                                                                                                                                                                                                                                                                                                                                                                                                              0.162
                                                                                                                                                                                                                                                                                                                                                                       9.620
4.497
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         0.049
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         0.105
                                                                                                                                                                                                                                                                                                                                                                        4.498
                                                                                                                                                                                                                                                                                                                                                                                                                              0.441
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         0.100
                                                                                                                                                                                                                                                                                                                                                                                  513
                                                                                                                                                                                                                                                                                                                                                                                                                              0.452
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        0.122
0.216
0.387
                                                                                                                                                                                                                                                                                                                                                                        4.210
                                                                                                                                                                                                                                                                                                                                                                                                                              0.462
                                                                                                                                                                                                                                                                                                                                                                        4.162
1.117
                                                                                                                                                                                                                                                                                                                                                                                                                             0.674
                                                                                                                                                                                                                                                                                                                                                                                                                               2.067
```

Then, using ROS_LOCALHOST_ONLY=1 and the cycloneDDS configuration @takam5f2 suggested, the performance can be improved again:

TOPIC NAME	HZ	AGE	SINCE LAST
/sensing/lidar/top/velodyne_packets	9.546	0.104	0.091
/sensing/lidar/top/pointcloud	9.725	0.141	0.055
/sensing/lidar/concatenated/pointcloud	9.749	0.147	0.047
/localization/util/measurement_range/pointcloud	9.825	0.164	0.031
/localization/util/voxel_grid_downsample/pointcloud	7.031	0.334	0.109
/localization/util/downsample/pointcloud	7.019	0.335	0.108
/localization/pose_estimator/pose_with_covariance	7.028	0.343	0.105
/perception/object_recognition/detection/apollo/labeled_clusters	6.017	0.357	0.051
/perception/object_recognition/detection/apollo/objects	6.022	0.501	0.072
<pre>/perception/object_recognition/detection/apollo/validation/objects</pre>	1.703	0.959	0.214
/perception/object_recognition/detection/objects	1.645	1.408	0.626
/perception/object_recognition/tracking/objects	1.646	1.418	0.608
/perception/object_recognition/objects	1.648	1.423	0.610

But still, it is far from being ideal. In particular voxel_grid_downsampling, labeled_clusters->objects->validation/objects and tracking processing are still very slow on my machine. But as far as I remember, these algorithms also existed in AAP (maybe simpler?)

Last but not least. I observe that despite all my effort configuring cycloneDDS, FastRTPS seems always way faster:

TOPIC NAME	1	HZ	T	AGE	ī	SINCE LAST
/sensing/lidar/top/velodyne_packets	1.	10.000	-1	0.104	- 1	0.001
/sensing/lidar/top/pointcloud	1.1	9.729	-1	0.140		0.052
/sensing/lidar/concatenated/pointcloud	1.0	0.749	-1	0.147	- 1	0.263
/localization/util/measurement_range/pointcloud	1.0	9.305	-1	0.167	- 1	0.035
/localization/util/voxel_grid_downsample/pointcloud	1.0	7.690	-1	0.321	- 1	0.111
/localization/util/downsample/pointcloud	i i	7.696	- İ	0.322	ΞÌ	0.110
/localization/pose_estimator/pose_with_covariance	i_	7.727	- İ	0.332	ΞÌ	0.093
/perception/object_recognition/detection/apollo/labeled_clusters	- i	0.754	- İ	0.326	ΞÌ	0.058
/perception/object_recognition/detection/apollo/objects	1.	0.700	-1	0.482	- 1	0.088
/perception/object_recognition/detection/apollo/validation/objects	1.0	0.585	1	0.910		0.303
/perception/object_recognition/detection/objects	1.0	0.593	-1	1.118	- 1	0.117
/perception/object_recognition/tracking/objects	Τi	0.593		1.125	Ξi	0.109
/perception/object_recognition/objects		0.593		1.129		0.104

I am wondering why FastRTPS is not recommended for autoware. Is it just faster in my setup? can I meet that performance with cycloneDDS with a better configuration?





takam5f2 on May 31, 2022 (Collaborator)

@VRichardJP

Thank you for sharing your experiment.

I suppose that another bottleneck like memory bandwidth or thread pool size limits Autoware's performance.

Could you show me your system configuration? Mine is here.

Resources	Description		
CPU physical Cores	8		
CPU logical Cores (threads)	16		
CPU Freq	3.3 GHz		
Memory size	32 GBytes		
Memory channel	Dual		
Memory speed	2666 GT/s		

I'd like to know the number of CPU logical core because it decides thread pool size. If it is less than 8, sensing pipeline might not work correctly.

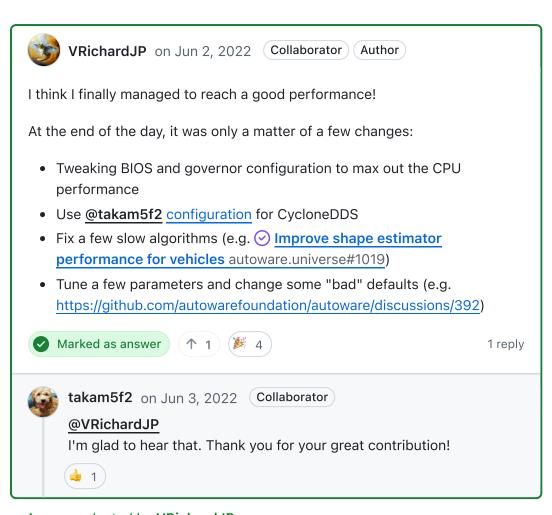
Sensing pipeline depends on memory speed (bandwidth), so I'd like to know them also.

Best,



VRichardJP on May 31, 2022 Collaborator Author

The machine I used in for this test has a i7-9700K (8 cores, 8 threads, 3.60GHz to 4.9GHz) and 32GB DDR4 @2666MHz. The computer I used to have for my AAP tests was slightly less powerful.



Answer selected by VRichardJP