Big latency jump over `/localization/util/voxel_grid_downsample/pointcloud` messages #2617

Answered by VRichardJP · **VRichardJP** asked this question in **Q&A**

VRichardJP on May 23, 2022 · Collaborator

I have remarked that messages over the `/localization/util/voxel_grid_downsample/pointcloud` topic are excessively old in my setup. My lidar processing pipeline is as follow:

1. `/sensing/lidar/top/velodyne_packets`
2. `/sensing/lidar/top/raw`
3. `/sensing/lidar/top/downsampled`
4. `/sensing/lidar/top/rectified`
5. `/sensing/lidar/top/pointcloud`
6. `/localization/util/measurement_range/pointcloud`
7. `/localization/util/voxel_grid_downsample/pointcloud`
8. `/localization/util/downsample/pointcloud`

I have written a simple script to track latency over these topics. Precisely, I track the average difference between the message stamp and the current clock over the last 10 messages. This method is appropriate on these topics since message stamp is preserved over the pipeline. Running the full stack `autoware.launch.xml`, I get the following results:

```
/sensing/lidar/top/velodyne_packets: 0.110
/sensing/lidar/top/downsampled/pointcloud_ex: 0.139
/sensing/lidar/top/rectified/pointcloud_ex: 0.140
/sensing/lidar/top/raw/pointcloud_ex: 0.154
/sensing/lidar/top/pointcloud: 0.253
/localization/util/measurement_range/pointcloud: 0.309
/localization/util/downsample/pointcloud: 1.206
/localization/util/voxel_grid_downsample/pointcloud: 1.221
```

For example here, message over `/sensing/lidar/top/velodyne_packets` topic are 110ms "late" on average. Similarly, messages over `/localization/util/measurement_range/pointcloud` topic are 309ms "late" on average. Because I take the average delay over the last 10 messages, some topic messages may appear older in earlier topic. But overall, the age increases as we go down the pipeline.

Category

🙏 Q&A

Labels

component:sensing

3 participants

The issue I have here is the huge age jump from `/localization/util/measurement_range/pointcloud` (309ms) to `/localization/util/voxel_grid_downsample/pointcloud` (1221ms). I am using the default `util.launch.py` so there is only one node in between: `VoxelGridDownsampleFilterComponent`

Not only latency gets high, but the drop rate becomes quite big as well. While messages on `/localization/util/measurement_range/pointcloud` are published at an average of 9.8Hz (my lidar sensor runs at 10Hz), almost 3/4 of the the messages are dropped by the `/localization/util/voxel_grid_downsample/pointcloud` topic:

```
$ ros2 topic hz
/localization/util/voxel_grid_downsample/pointcloud
average rate: 2.720
        min: 0.235s max: 0.574s std dev: 0.12769s window: 4
average rate: 2.823
        min: 0.235s max: 0.574s std dev: 0.09628s window: 8
average rate: 2.763
        min: 0.212s max: 0.620s std dev: 0.12277s window: 11
```

I have tracked the execution time of `VoxelGridDownsampleFilterComponent`. `VoxelGridDownsampleFilterComponent::filter()` takes around 300ms to execute, and `Filter::input_indices_callback` does no pre/postprocessing, so total execution time is the same. 300ms execution time is not particularly fast, but that is very far from the 900ms added latency I found. So where are the extra 600ms coming from?

Just to make sure, I have verified that the voxel grid downsample component is running in the same container than other lidar components:

```
$ ros2 component list
/sensing/lidar/top/pointcloud_preprocessor/velodyne_node_conta...
  1  /sensing/lidar/top/velodyne_driver
  2  /sensing/lidar/top/velodyne_convert_node
  3  /localization/util/crop_box_filter_measurement_range
  4  /sensing/lidar/top/fixed_downsample_filter
  5  /localization/util/voxel_grid_downsample_filter
  6  /sensing/lidar/top/velodyne_interpolate_node
  7  /localization/util/random_downsample_filter
  8  /sensing/lidar/top/crop_box_filter_self
```

...and that the QoS configuration is correct:

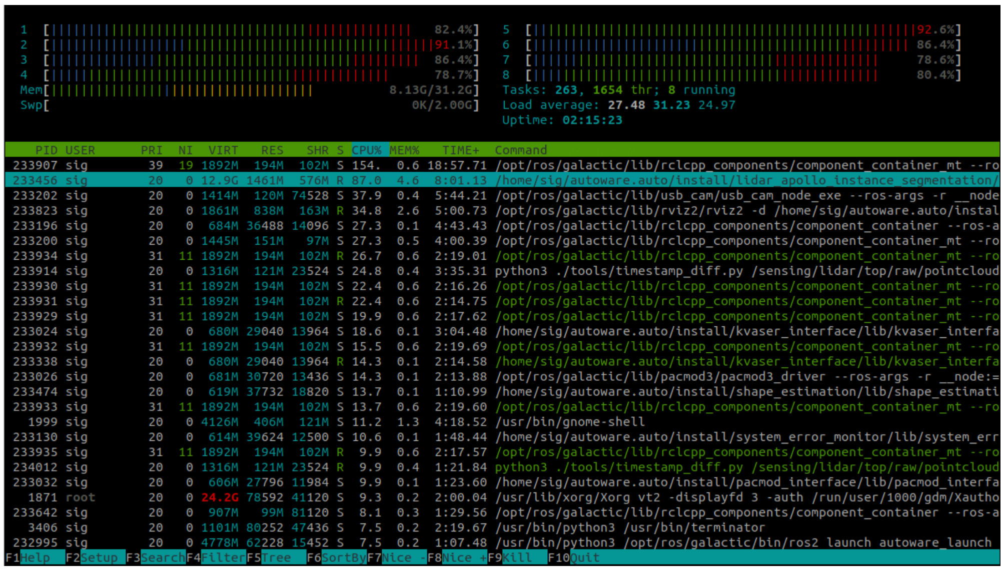
```
$ ros2 topic info
/localization/util/voxel_grid_downsample/pointcloud -v
Type: sensor_msgs/msg/PointCloud2

Publisher count: 1

Node name: voxel_grid_downsample_filter
Node namespace: /localization/util
Topic type: sensor_msgs/msg/PointCloud2
Endpoint type: PUBLISHER
GID:
f5.3a.10.01.cb.ad.2f.89.88.17.a3.04.00.00.6e.03.00.00.00.00.00.00.0
```

```
QoS profile:
  Reliability: BEST_EFFORT
  Durability: VOLATILE
  Lifespan: 9223372036854775807 nanoseconds
  Deadline: 9223372036854775807 nanoseconds
  Liveliness: AUTOMATIC
  Liveliness lease duration: 9223372036854775807 nanoseconds

Subscription count: 1

Node name: random_downsample_filter
Node namespace: /localization/util
Topic type: sensor_msgs/msg/PointCloud2
Endpoint type: SUBSCRIPTION
GID:
f5.3a.10.01.cb.ad.2f.89.88.17.a3.04.00.00.9b.04.00.00.00.00.00.00.0
QoS profile:
  Reliability: BEST_EFFORT
  Durability: VOLATILE
  Lifespan: 9223372036854775807 nanoseconds
  Deadline: 9223372036854775807 nanoseconds
  Liveliness: AUTOMATIC
  Liveliness lease duration: 9223372036854775807 nanoseconds
```

Nothing particularly stands out to me, and yet I mysteriously have an extra delay of 600ms over that topic.

Finally, I have checked how that latency evolves with CPU usage. In the setup I have described so far, my CPU is always at 100%. Obviously, having 100% CPU usage is bound to have an impact on performance. But if that was just a raw performance issue, I would expect all the modules to suffer more or less equally. In my situation, the voxel grid downsample component is outstandingly slow compared to other components. Moreover, I would expect NICE-ing the component processes would have a positive impact on latency, but it does not.

If I kill a few unrelated processes (apollo, rviz...) I can manage to reduce CPU usage to 80%:



```
  1  [||||||||||||||||||||||||||||||||||||||||||||||||     82.4%]   5  [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||92.6%]
  2  [|||||||||||||||||||||||||||||||||||||||||||||||||||||91.1%]   6  [|||||||||||||||||||||||||||||||||||||||||||||||||||    86.4%]
  3  [|||||||||||||||||||||||||||||||||||||||||||||||      86.4%]   7  [|||||||||||||||||||||||||||||||||||||||||||||          78.6%]
  4  [|||||||||||||||||||||||||||||||||||||||||||          78.7%]   8  [|||||||||||||||||||||||||||||||||||||||||||||          80.4%]
  Mem[||||||||||||||||||||||||||||||||||||         8.13G/31.2G]   Tasks: 263, 1654 thr; 8 running
  Swp[                                               0K/2.00G]   Load average: 27.48 31.23 24.97
                                                                 Uptime: 02:15:23

    PID USER       PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 233907 sig         39  19 1892M  194M  102M S 154.  0.6 18:57.71 /opt/ros/galactic/lib/rclcpp_components/component_container_mt --ro
 233456 sig         20   0 12.9G 1461M  576M R 87.0  4.6  8:01.13 /home/sig/autoware.auto/install/lidar_apollo_instance_segmentation/
 233202 sig         20   0 1414M  120M 74528 S 37.9  0.4  5:44.21 /opt/ros/galactic/lib/usb_cam/usb_cam_node_exe --ros-args -r __node
 233823 sig         20   0 1861M  838M  163M R 34.8  2.6  5:00.73 /opt/ros/galactic/lib/rviz2/rviz2 -d /home/sig/autoware.auto/instal
 233196 sig         20   0  684M 36488 14096 S 27.3  0.1  4:43.43 /opt/ros/galactic/lib/rclcpp_components/component_container --ros-a
 233200 sig         20   0 1445M  151M   97M S 27.3  0.5  4:00.39 /opt/ros/galactic/lib/rclcpp_components/component_container_mt --ro
 233934 sig         31  11 1892M  194M  102M R 26.7  0.6  2:19.01 /opt/ros/galactic/lib/rclcpp_components/component_container_mt --ro
 233914 sig         20   0 1316M  121M 23524 S 24.8  0.4  3:35.31 python3 ./tools/timestamp_diff.py /sensing/lidar/top/raw/pointcloud
 233930 sig         31  11 1892M  194M  102M S 22.4  0.6  2:16.26 /opt/ros/galactic/lib/rclcpp_components/component_container_mt --ro
 233931 sig         31  11 1892M  194M  102M R 22.4  0.6  2:14.75 /opt/ros/galactic/lib/rclcpp_components/component_container_mt --ro
 233929 sig         31  11 1892M  194M  102M S 19.9  0.6  2:17.62 /opt/ros/galactic/lib/rclcpp_components/component_container_mt --ro
 233024 sig         20   0  680M 29040 13964 S 18.6  0.1  3:04.48 /home/sig/autoware.auto/install/kvaser_interface/lib/kvaser_interfa
 233932 sig         31  11 1892M  194M  102M S 15.5  0.6  2:19.69 /opt/ros/galactic/lib/rclcpp_components/component_container_mt --ro
 233338 sig         20   0  680M 29040 13964 R 14.3  0.1  2:14.58 /home/sig/autoware.auto/install/kvaser_interface/lib/kvaser_interfa
 233026 sig         20   0  681M 30720 13436 S 14.3  0.1  2:13.88 /opt/ros/galactic/lib/pacmod3/pacmod3_driver --ros-args -r __node:=
 233474 sig         20   0  619M 37732 18820 S 13.7  0.1  1:10.99 /home/sig/autoware.auto/install/shape_estimation/lib/shape_estimati
 233933 sig         31  11 1892M  194M  102M S 13.7  0.6  2:19.60 /opt/ros/galactic/lib/rclcpp_components/component_container_mt --ro
   1999 sig         20   0 4126M  406M  121M S 11.2  1.3  4:18.52 /usr/bin/gnome-shell
 233130 sig         20   0  614M 39624 12500 S 10.6  0.1  1:48.44 /home/sig/autoware.auto/install/system_error_monitor/lib/system_err
 233935 sig         31  11 1892M  194M  102M R  9.9  0.6  2:17.57 /opt/ros/galactic/lib/rclcpp_components/component_container_mt --ro
 234012 sig         20   0 1316M  121M 23524 R  9.9  0.4  1:21.84 python3 ./tools/timestamp_diff.py /sensing/lidar/top/raw/pointcloud
 233032 sig         20   0  606M 27796 11984 S  9.9  0.1  1:23.60 /home/sig/autoware.auto/install/pacmod_interface/lib/pacmod_interfa
   1871 root        20   0 24.2G 78592 41120 S  9.3  0.2  2:00.04 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xautho
 233642 sig         20   0  907M   99M 81120 S  8.1  0.3  1:29.56 /opt/ros/galactic/lib/rclcpp_components/component_container --ros-a
   3406 sig         20   0 1101M 80252 47436 S  7.5  0.2  2:19.67 /usr/bin/python3 /usr/bin/terminator
 232995 sig         20   0 4778M 62228 15452 S  7.5  0.2  1:07.48 /usr/bin/python3 /opt/ros/galactic/bin/ros2 launch autoware_launch
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

In that situation, I would expect the CPU not to be a bottleneck anymore. But still, I observe the latency jump on the `/localization/util/voxel_grid_downsample/pointcloud:` topic:

```
/sensing/lidar/top/velodyne_packets: 0.103
/sensing/lidar/top/raw/pointcloud_ex: 0.118
/sensing/lidar/top/downsampled/pointcloud_ex: 0.119
/sensing/lidar/top/rectified/pointcloud_ex: 0.122
/sensing/lidar/top/pointcloud: 0.150
/localization/util/measurement_range/pointcloud: 0.165
/localization/util/voxel_grid_downsample/pointcloud: 0.707
/localization/util/downsample/pointcloud: 0.707
```

Now I have 707-165 = 542ms delay on the voxel grid downsample component. But the `input_indices_callback` function execution time is now around 100ms. Once again, I mysteriously lose more than 400ms but it seems not related to computation. It is better than previous delay (600ms), but it still bad. The message frequency is also better (7.5Hz) but there are still many messages being dropped.

As a final test, I have killed everything but the `velodyne_node_container`. This time I get good results:

```
/sensing/lidar/top/velodyne_packets: 0.103
/sensing/lidar/top/downsampled/pointcloud_ex: 0.111
/sensing/lidar/top/raw/pointcloud_ex: 0.113
/sensing/lidar/top/rectified/pointcloud_ex: 0.132
/sensing/lidar/top/pointcloud: 0.138
/localization/util/measurement_range/pointcloud: 0.149
/localization/util/voxel_grid_downsample/pointcloud: 0.209
/localization/util/downsample/pointcloud: 0.209
```

The extra latency from `/localization/util/measurement_range/pointcloud` to `/localization/util/voxel_grid_downsample/pointcloud` is around 60ms, which is the `filter()` function execution time.

I have yet to understand what causes my 400ms~600ms ghost latency. If it seems not to be entirely related to cpu load since I still have the issue when the CPU usage is <100%. But then, where does it come from? poor inter-process communication? Cyclone DDS limitation?

Anyone has an idea?

↑ 1

Answered by **VRichardJP** on May 24, 2022

In the end, my problem was very simple: the filters default QoS configuration is to keep the last 5 messages. Since the `voxel_grid_downsample_filter` component is slower than `crop_box_filter_measurement_range`, the messages from `/localization/util/measurement_range/pointcloud` topic

View full answer ↓

3 comments · 2 replies

Oldest Newest Top

VRichardJP on May 24, 2022 Collaborator Author

In the end, my problem was very simple: the filters default QoS configuration is to keep the last 5 messages. Since the `voxel_grid_downsample_filter` component is slower than `crop_box_filter_measurement_range`, the messages from `/localization/util/measurement_range/pointcloud` topic accumulate and fill the input queue of `voxel_grid_downsample_filter`. So in order for the latter to process the latest pointcloud and publish it to `/localization/util/voxel_grid_downsample/pointcloud`, the component needs to first process the 4 messages before. This caused the big latency I was observing.

Since what matters is the component processing speed relative to others, I think that problem is not restricted to my setup and the effect should be visible on faster machine too.

Then, what is the most appropriate change? change all filters queue size to 1?

✓ Marked as answer ↑ 1 1 reply

miursh on May 31, 2022 Collaborator

My personal opinion is that the queue size should be 1 in a real-time system, unless the node itself has a message buffer.
If some node runs in different (fast) frequency, the others should have message buffer.

Answer selected by **VRichardJP**

nainaigetuide on Nov 1, 2022

@VRichardJP How to change the QOS configuration to 1?

↑ 1 1 reply

VRichardJP on Nov 2, 2022 Collaborator Author

This is the parameter to change
https://github.com/autowarefoundation/autoware.universe/blob/4c0f75f755ea3596401e39c390db75b7a0676842/sensing/pointcloud_preprocessor/src/filter.cpp#L72