Put every node into a single component container? #3818

Unanswered xmfcx asked this question in Ideas

xmfcx on Sep 8, 2023 Maintainer

https://docs.ros.org/en/humble/Tutorials/Intermediate/Composition.html

Component containers removes the network overhead between nodes by using intra-process communication. Thus it makes the overall system more efficient.

When running AWSIM demo, there are 16 component containers. I think this is too many, especially considering the containers with only a single node loaded.

```
mfc@mfc-leo:~$ ros2 component list
/autoware_api/external/rtc_controller/container
  1  /autoware_api/external/rtc_controller/node
/map/map_container
  1  /map/lanelet2_map_loader
  2  /map/lanelet2_map_visualization
  3  /map/pointcloud_map_loader
  4  /map/vector_map_tf_generator
/system/mrm_emergency_stop_operator/mrm_emergency_stop_operator_con
  1  /system/mrm_emergency_stop_operator
/system/component_state_monitor/container
  1  /system/component_state_monitor/component
/perception/traffic_light_recognition/traffic_light_node_container
  1  /perception/traffic_light_recognition/traffic_light_classifier
  2  /perception/traffic_light_recognition/traffic_light_image_decor
  3  /perception/traffic_light_recognition/crosswalk_traffic_light_
  4  /perception/traffic_light_recognition/traffic_light_roi_visual
/planning/scenario_planning/lane_driving/behavior_planning/behavior_
  1  /planning/scenario_planning/lane_driving/behavior_planning/beha
  2  /planning/scenario_planning/lane_driving/behavior_planning/beha
/planning/scenario_planning/parking/parking_container
  1  /planning/scenario_planning/parking/costmap_generator
  2  /planning/scenario_planning/parking/freespace_planner
/sensing/lidar/left/pointcloud_preprocessor/velodyne_node_container
  1  /sensing/lidar/left/pointcloud_distortion_relay
  2  /sensing/lidar/left/crop_box_filter_self
  3  /sensing/lidar/left/crop_box_filter_mirror
  4  /sensing/lidar/left/ring_outlier_filter
/control/control_container
  1  /control/external_cmd_converter
  2  /control/external_cmd_selector
  3  /control/trajectory_follower/controller_node_exe
  4  /control/trajectory_follower/lane_departure_checker_node
  5  /control/shift_decider
  6  /control/vehicle_cmd_gate
  7  /control/operation_mode_transition_manager
/perception/object_recognition/detection/clustering/euclidean_clust
  1  /perception/object_recognition/detection/clustering/short_dist
```

Category

Ideas

Labels

None yet

5 participants

```
  2  /perception/object_recognition/detection/clustering/voxel_grid_
  3  /perception/object_recognition/detection/clustering/long_distan
  4  /perception/object_recognition/detection/clustering/outlier_fi
  5  /perception/object_recognition/detection/clustering/concat_dowr
  6  /perception/object_recognition/detection/clustering/euclidean_
/default_ad_api/container
  1  /default_ad_api/node/autoware_state
  2  /default_ad_api/node/fail_safe
  3  /default_ad_api/node/interface
  4  /default_ad_api/node/localization
  5  /default_ad_api/node/motion
  6  /default_ad_api/node/operation_mode
  7  /default_ad_api/node/planning
  8  /default_ad_api/node/routing
/system/mrm_comfortable_stop_operator/mrm_comfortable_stop_operator_
  1  /system/mrm_comfortable_stop_operator
/sensing/lidar/right/pointcloud_preprocessor/velodyne_node_containe
  1  /sensing/lidar/right/pointcloud_distortion_relay
  2  /sensing/lidar/right/crop_box_filter_self
  3  /sensing/lidar/right/crop_box_filter_mirror
  4  /sensing/lidar/right/ring_outlier_filter
/planning/scenario_planning/lane_driving/motion_planning/motion_plar
  1  /planning/scenario_planning/lane_driving/motion_planning/obstac
  2  /planning/scenario_planning/lane_driving/motion_planning/obstac
  3  /planning/scenario_planning/lane_driving/motion_planning/surro
  4  /planning/scenario_planning/lane_driving/motion_planning/obstac
/pointcloud_container
  1  /localization/util/crop_box_filter_measurement_range
  2  /sensing/lidar/concatenate_data
  3  /perception/object_recognition/detection/voxel_based_compare_ma
  4  /perception/occupancy_grid_map/occupancy_grid_map_node
  5  /perception/obstacle_segmentation/crop_box_filter
  6  /localization/util/voxel_grid_downsample_filter
  7  /perception/obstacle_segmentation/common_ground_filter
  8  /localization/util/random_downsample_filter
  9  /perception/obstacle_segmentation/occupancy_grid_map_outlier_f
/sensing/lidar/top/pointcloud_preprocessor/velodyne_node_container
  1  /sensing/lidar/top/pointcloud_distortion_relay
  2  /sensing/lidar/top/crop_box_filter_self
  3  /sensing/lidar/top/crop_box_filter_mirror
  4  /sensing/lidar/top/ring_outlier_filter
```

Why don't we put all nodes into a single `autoware_component_container`?

What are the downsides of such a change?

I can list some pros and cons of such an action:

Pros

- **Reduced Inter-Process Communication (IPC) Overhead:** With all nodes running in a single process, the need for inter-process communication would be nearly eliminated. This can improve performance, especially when there is a high frequency of messages between nodes. This will help reduce the potential additional bandwidth when we introduce heartbeats and monitoring to all of the nodes.

- **Simplified Management:** With a single container to manage, starting, stopping, and monitoring the system can become simpler.

- **Memory Efficiency:** Running all nodes in a single process can sometimes result in better memory usage, as there's only one instance of the ROS2 runtime and potentially fewer duplicated resources.

Cons

- **Single Point of Failure:** If one node crashes or there's an error in the process, it can potentially bring down all nodes in the container.
  - I think this is a good opportunity to create a parallel system which is much simpler than Autoware and focuses on emergency behaviors. But needs to be discussed more.
- **Resource Contention:** All nodes in the same process would share the same memory and CPU resources. If one node becomes resource-hungry, it could negatively impact the performance of other nodes.
  - I think this won't make much of a difference since they all contend for resources regardless anyways. I might be missing something too.
- **Complexity:** Merging all nodes into a single container might initially seem like a simplification, but it can introduce complexity when trying to isolate specific nodes for debugging or when adding new nodes to the system.
  - Debugging Autoware right now is already hard right now. And even in current state, we need to disable containers for certain nodes to debug anyway.

I would like to hear your opinions about combining all into a single component container too.

cc. @VRichardJP @mitsudome-r @TakaHoribe @miursh @isamu-takagi @yukkysaito @kaancolak @mehmetdogru @esteve

↑ 1 👍 3

2 comments · 8 replies

Oldest | Newest | Top

yukkysaito on Sep 8, 2023 Maintainer

Thank you for information 👍
From a safety standpoint, it would be better to separate them as much as possible.
If too many nodes are grouped together in a container, only the MRM of emergency shutdown will be triggered when a failure occurs.
If the containers are separated, it is possible to pull over in the event of a failure.

However, it is difficult to decide which is better, as it depends on the product requirements as to what kind of MRM you want to trigger at what time.

↑ 1 👍 1 6 replies

Show 1 previous reply

xmfcx on Sep 8, 2023 Maintainer Author

Like, even if planning nodes crash, we should be able to do pullover maneuvers safely.

isamu-takagi on Sep 8, 2023 Maintainer

I think this choice should be made available to Autoware users. Since ROS provides a component container, this can be done by only changing the launch file. For this, Autoware needs to provide all nodes as components.

Then the only consideration is which configuration of launch files autoware_launch provides by default.

xmfcx on Sep 8, 2023 Maintainer Author edited

@isamu-takagi Sure, we can allow any configuration. I'm trying to figure out if there is a single, common good way of doing it.

I agree we should make all the nodes into composable nodes as a goal.

But this discussion is about whether or not we should focus our attention to be single component container approach.

Like, we could put all the existing composable nodes into a single container today. But I want to learn about the arguments against this.

👍 1

esteve on Sep 8, 2023 Maintainer

> @xmfcx I agree we should make all the nodes into composable nodes as a goal.

I believe we already made all nodes composable, I recall checking every node and updating them to be composable. Not sure if all the nodes have followed that pattern since then. In any case, as @yukkysaito pointed out, if all nodes are composable, it's only a matter of changing the launch files to put them in the same component manager.

From a safety standpoint, it's best to keep them as separate processes IMHO, if a component crashes, it may bring down the entire component manager and the rest of the components. If we want to improve communication between nodes, we can look into the loaned messages API (see autowarefoundation/autoware.universe#1794) and rmw_iceoryx

👍 1

xmfcx on Sep 11, 2023 Maintainer Author

> If we want to improve communication between nodes, we can look into the loaned messages API (see autowarefoundation/autoware.universe#1794) and rmw_iceoryx

This is the best way to achieve this but it is also a long and time consuming task.

Thanks for sharing your concerns.

👍 1

VRichardJP on Sep 8, 2023 Collaborator

Here's my two cents:

> Reduced Inter-Process Communication (IPC) Overhead

I would like to see the numbers on this. IPC shines when transferred data is heavy (point cloud, image). I think today most of the components manipulating heavy data are already together. I am not sure there is any significant performance improvement when exchanged data is small.

> Memory Efficiency

Same, I would like to see numbers here too. For instance, if you run autoware twice, once with the current components, once with everything in one container, do you see a significant difference in memory usage?

> Single Point of Failure

This is a big issue, because crash happen, and won't disappear anytime soon. If the gnss poser crashes, you still want lidar localization to work, planning to find safe trajectories and emergency module to stop the vehicle when necessary.

↑ 1 👍 1 2 replies

xmfcx on Sep 11, 2023 Maintainer Author

@VRichardJP I agree with you, I think the same way about only using component containers for heavy messages.

But I also have concerns about having too many small messages too. I am working on: feat: add autoware_node and autoware_control_center and I didn't benchmark the capabilities of DDS when there are so many and frequent heartbeat messages sent around.

I also wouldn't move on with this without doing some benchmarks. I wanted to see if I was missing something obvious and developers' thoughts and past experiences about the topic.

Also I wanted to know about how we approach the safety issues in general.

Like, when a node crashes, even an insignificant one, do we still want everything else to operate? Or do we just want to send an emergency stop signal?

Or, in what scenarios we want an MRM Pullover to activate? (It is not implemented yet and TIER IV devs are working on this feature)

Maybe we could have a small, simpler, lighter safety version of Autoware running in parallel and would handle safety cases when the main Autoware fails.

- Right now this is not planned, current expectation is that if planner component fails, the vehicle should do emergency stop.

Anyways, this thread was to share our concerns for such scenario. I'm happy for everyone who participated and would like to learn more about your perspectives on how we can handle the safety issues. This includes:

- When a sensor crashes
- When a sensor is occluded
- When a node crashes
- When a node becomes irresponsive
- How to categorize the priorities for nodes (requirement/necessity levels for self-driving)

VRichardJP on Sep 11, 2023 Collaborator edited

> Maybe we could have a small, simpler, lighter safety version of Autoware running in parallel and would handle safety cases when the main Autoware fails.

This is a common approach I have seen in other AD stacks, where a "safety" stack has limited features and can override the "best effort" stack at any time. However, both stacks have very different requirements, and I would not see Autoware being anything but the "best effort" stack.