Autoware Transform Listener - performance improvement #5385

New issue

Open 8 of 13 tasks **amadeuszsz** opened this issue 3 weeks ago · 12 comments

amadeuszsz commented 3 weeks ago • edited Contributor

Checklist

- [x] I've read the contribution guidelines.
- [x] I've searched other issues and no duplicate issues were found.
- [x] I've agreed with the maintainers that I can plan this task.

Description

Autoware runtime characterizes with hundreds nodes running at the same time. Each node uses Pub / Sub mechanism for communication. One of the most common subscribed topic is `/tf` and `/tf_static`. These topics are subscribed implicitly while constructing `TransformListener` object - after initialization a new node `called transform_listener_impl_xxxxxxxxxxx` appears in nodes graph. For `/tf` topic there is not much what could be done - ROS node needs an update for changing vehicle's ego pose. For `/tf_static` it seems there is no need for continuous subscription, but ROS 2 API does not allow for controlling listener behavior.

In Autoware, most of nodes need a transformation between sensor kit links. By default approach we subscribe `/tf` topic even if it is not necessary. These extra nodes with high frequency `/tf` topic subscription increase CPU utilization significantly.

First, let's clarify two definitions:

Static Transform Listener - a listener which reads requested transformations, save it to internal buffer, and stop subscribing topics. Next requests for same transformations just reads transformation from internal buffer.

Assignees

amadeuszsz

Labels

component:sensing

type:performance

Projects

Sensing & Perception Worki...

Status: In Progress +1 more

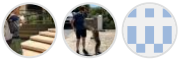
Milestone

No milestone

Development

No branches or pull requests

3 participants



Dynamic Transform Listener - a listener with default ROS 2 behavior, always reads recent transformations, regardless if it is `/tf` or `/tf_static`.

Recently ROS 2 Rolling got new feature which stands for Static Transform Listener. PR provides separate class for listener which subscribes only one topic - `/tf_static`. This solution potentially relaxes CPU load, but static TF listener has to be defined explicitly. Additionally, extra listener node will still remain in node graph. Nevertheless, this feature is not available in Humble or Jazzy distribution neither.

Purpose

Reduce CPU load by getting rid of unnecessary TF listeners in ROS nodes graph.

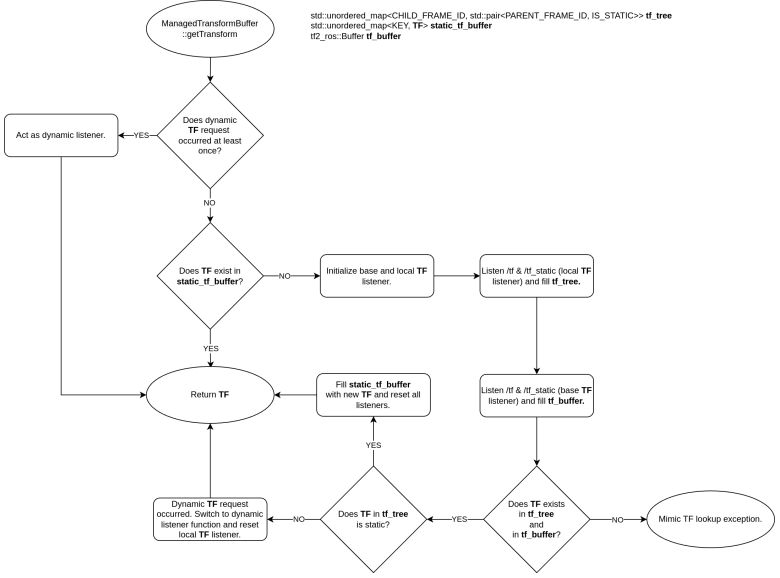
Possible approaches

Current solution - explicitly defined type of listener

The solution is simple - it wrapping listener and buffer to a single class and explicitly declares if incoming transforms are static or dynamic. Unfortunately, we encountered and issue and all nodes which use this feature got has_static_tf_only parameter false by default. The next step would be populate to every single launch file and config file a parameter has_static_tf_only with appropriate value. This is possible, but based on number of Pilot.Auto repositories it will bring headaches. Secondly, exposing this parameter is likely to be bug-prone for the community and may sometimes lead to difficult to find sources of future issues.

Possible solution - implicitly defined type of listener

What if API does not expose any parameter and decide itself to behave like dynamic or static transform listener?



The diagram seems complicated, but this applies only for first TF requests. We can differentiate few behaviors for **n_static** occurrences of specific static TF and **n_dynamic** occurrences of specific dynamic TF:

n_static=1, **n_dynamic**=0 - initializes TF listeners, fills `static_tf_buffer` with resultant TF, reset TF listeners and returns TF.

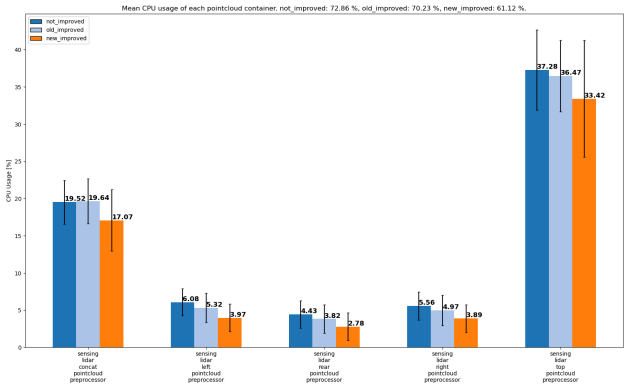
n_static>1, **n_dynamic**=0 - reads TF from `static_tf_buffer` and returns it.

n_static>=0, **n_dynamic**=1 - initializes TF listeners, overrides listener function to dynamic listener (default ROS 2 approach), returns TF.

n_static>=0, **n_dynamic**>1 - behaves like a default ROS 2 TF listener.

First request might take slightly more time - it depends on request timeout (exposed as an argument in function call same way as it is done for lookupTransform). However, in general it will benefit with significant performance improvement during Autoware runtime. What is important here, **if ROS node which use this class encounters dynamic TF, getTransform function will be overridden and starts behaving like a ROS-ish TF listener**.

The CPU load comparison confirms performance improvement.



This improvement is caused by reduced number of TF listeners in nodes containers.

	Not improved	Old improved	New improved
top	3	2	0
left	3	2	0
right	3	2	0
rear	3	2	0
concat	1	1	0
total	13	9	0

Old improved implementation by default does the difference only if `target_frame` = `source_frame`. It can achieve same performance as new improved implementation, but this will require to set `has_static_tf_only` parameter to True.

The proposed approach might fail in one scenario - static transformations change during the runtime. For now, Autoware does not consist of such a behavior. However, dynamic sensor calibration feature might introduce this problem. In that case, an extra service server for resetting `static_tf_buffer` could handle this.

Another point is why we need `static_tf_buffer` at all and traversal function, which checks whether TF is static and fills this local buffer with that information. It could be integrated directly into casual ROS-ish TF buffer, but due to current `geometry2` design, it would require big API changes. Without that changes and having current design in mind:

```
tf_buffer_ = std::make_shared<tf2_ros::Buffer>
tf_listener_ = std::make_shared<tf2_ros::Trans
```

we could expose call `tf_buffer_->wereStaticTFsOnlyRequested()` and turning off / on `tf_listener_` easily. We would need just a TF tree placed somewhere in tf2 buffer - parent & child frame and `is_static` information are already there. Nevertheless, it needs ROS 2 core changes and might be difficult to merge in short period of time.

Definition of done

- [x] New TF listener design
- [x] Proposed design implementation
- [x] Unit tests
- [x] Apply changes for packages which already use old improved version (`autoware_pointcloud_preprocessor` and `autoware_ground_segmentation`)
- [x] Performance tests for `autoware_pointcloud_preprocessor` node container
- [ ] Apply changes across Autoware packges which use default ROS 2 TF listener
  - [ ] Package A
  - [ ] Package B
  - [ ] ...
- [ ] Performance tests for Autoware sample rosbag demo

👍 1

amadeuszsz added component:sensing type:performance labels 3 weeks ago

amadeuszsz self-assigned this 3 weeks ago

amadeuszsz added this to **Sensing & Perception Working Group** 3 weeks ago

github-project-automation bot moved this to **To triage** in **Sensing & Perception Working Group** 3 weeks ago

amadeuszsz mentioned this issue 3 weeks ago

perf(autoware_universe_utils): introduce managed transform buffer with implicitly defined listener type Open

autowarefoundation/autoware.universe#9197

 **amadeuszsz** moved this from **To triage** to **In Progress** in **Sensing & Perception Working Group** 3 weeks ago

amadeuszsz commented 3 weeks ago — Contributor — Author

@yukkysaito could I ask you for your comments? Also I would appreciate if you could take a look into linked PR 🙏

yukkysaito commented 2 weeks ago — Contributor

@amadeuszsz cc @mitsudome-r Thank you for your proposal.

I think it's an excellent suggestion, but I have three questions as a maintainer.

First, you mentioned that the new feature, "Static Transform Listener," has already been introduced in ROS 2 Rolling. Would it be possible to register this package in the autoware.repos so that it could also be used in Humble?

Second, if we are able to utilize the Static Transform Listener in ROS 2 Rolling, what would be the advantages and disadvantages of using this feature? Would the proposed feature still offer better performance? Unless there is a specific reason, reinventing the wheel from a maintenance perspective isn't ideal, as unique implementations can make Autoware more challenging for users.

Third, if the response to the second question suggests that this proposal is indeed better, would it be possible to present this idea to the ROS 2 community?

amadeuszsz commented 2 weeks ago — Contributor — Author

@yukkysaito thank you for answer!

> First, you mentioned that the new feature, "Static Transform Listener," has already been introduced in ROS 2 Rolling. Would it be possible to register this package in the autoware.repos so that it could also be used in Humble?

This class resides in one of the common headers, therefore no extra package is needed. Putting `geometry2` into `autoware.repos` and rebuilding this package from source will cause linker issues since many packages depends on this single header. In a result, e.g. `rviz2` will crash. I assume rebuilding half of ROS from source is not the best idea for us.

> Second, if we are able to utilize the Static Transform Listener in ROS 2 Rolling, what would be the advantages and disadvantages of using this feature?

Rolling implementation requires explicitly defining type of listener. In other words, we need to define our listener as an instance of `TransformListener` or `StaticTransformListener`. In Autoware, usually we exposing frames as a parameters or read them from header, therefore we need a TF listener able to handle this automatically. Another thing is that for Rolling implementation the listener will remain as a node in ROS graph. After running Autoware we will end up with dozens of `transform_listener_impl_xxxxxxxxxxx` nodes from `StaticTransformListener` class. However, it has probably negligible performance impact.

> Would the proposed feature still offer better performance?

If somehow we can use `StaticTransformListener` (not available in humble, iron and jazzy) and correctly populate this change to all Autoware nodes, no performance differences. However, frames for us are parameters and we can't be 100 % sure if TF in that part of code is static or not. Also from user perspective is easier to don't think about it if it can be done automatically.

> Unless there is a specific reason, reinventing the wheel from a maintenance perspective isn't ideal, as unique implementations can make Autoware more challenging for users.

We already use something custom in our nodes. `ManagedTransformListener` same way wraps default `TransformListener`.

> Third, if the response to the second question suggests that this proposal is indeed better, would it be possible to present this idea to the ROS 2 community?

Good point. If we could apply changes directly into core ROS, I would make this implementation clearer by expanding tf2 API (`geometry2` repository). In best case scenario we might have this feature from ROS 2 K*.

If we merge it, we could go for ROS 2 core changes. If it will become accepted and available in one of future distros, we remove our algorithm implementation and keep API. This way our class will expand base class by only different return type so it will act as a simple wrapper - no API changes, no difference from user point of view.

mitsudome-r commented last week Member

> Good point. If we could apply changes directly into core ROS, I would make this implementation clearer by expanding tf2 API (geometry2 repository). In best case scenario we might have this feature from ROS 2 K*.

I like this idea. If we are making changes to the use of tf in Autoware, I think we should consider feeding that back to ROS 2 upstream rather than keeping in our own code. Of course, we might have to keep the code in Autoware repository for a while until it becomes available, but I would like to have a plan to merge it to ROS 2.

We can also consider backporting it to Humble/Jazzy once it is merged to rolling.

mitsudome-r commented last week Member

> First request might take slightly more time - it depends on request timeout (exposed as an argument in function call same way as it is done for lookupTransform)

Just as a minor feedback, we might want to also consider adding a parameter like "force_dynamic" in constructor for people who might want to use it like default TF Listener without changing the code to improve the performance for the first time.

amadeuszsz commented last week Contributor Author

Thank you **@mitsudome-r** for your feedback.

> I like this idea. If we are making changes to the use of tf in Autoware, I think we should consider feeding that back to ROS 2 upstream rather than keeping in our own code. Of course, we might have to keep the code in Autoware repository for a while until it becomes available, but I would like to have a plan to merge it to ROS 2.
>
> We can also consider backporting it to Humble/Jazzy once it is merged to rolling.

This would be perfect. Attached PR consist of implementation which will be reduced significantly after merging this feature into core of ROS. Declarations in header of public methods will stay same - no next refactor needed.

> Just as a minor feedback, we might want to also consider adding a parameter like "force_dynamic" in constructor for people who might want to use it like default TF Listener without changing the code to improve the performance for the first time.

We already have managed parameter with default value `managed=true`. Of course we can change naming convention.

❤️ 1

yukkysaito commented last week • edited Contributor

@amadeuszsz Thank you for your response.

Overall, I think it looks good. If it can be implemented without dependency on Autoware, it might be better to manage the package in a separate repository and register it as a release repository.

I will create a separate repository, so could you place the package there?

👍 1

amadeuszsz commented last week Contributor Author

@yukkysaito

> If it can be implemented without dependency on Autoware, it might be better to manage the package in a separate repository and register it as a release repository.
>
> I will create a separate repository, so could you place the package there?

Sure! Please, share the link.
From my perspective (I believe for CI too), it makes no difference where we place this feature, but I have one question - what is the motivation for placing it out of autoware.universe? Before I assumed that our goal is to reduce packages in `autoware.repos`, but I might be wrong 🤷

yukkysaito commented yesterday Contributor

@amadeuszsz Sorry for the delay. I'll create it today and send the link.

The purpose is that separating the packages to be registered in the ROS 2 repository from those not to be registered makes the process easier.

👍 1

yukkysaito commented yesterday Contributor

@amadeuszsz
Could you submit a PR for this? You also have permission to merge it.

https://github.com/autowarefoundation/ManagedTransformListener

amadeuszsz commented yesterday • edited Contributor Author

> @amadeuszsz Could you submit a PR for this? You also have permission to merge it.
>
> https://github.com/autowarefoundation/ManagedTransformListener

Yes, but the class is called `ManagedTransformBuffer`. Do you want to rename class from `ManagedTransformBuffer` to `ManagedTransformListener`? I called it `ManagedTransformBuffer` since this class mimics `tf2_ros::TransformBuffer` API, listener is hidden inside implementation.

Maybe I provided some ambiguity using "managed listener" / "managed buffer" interchangeably in this thread, sorry. From code point of view this is buffer.

yukkysaito commented 9 hours ago Contributor

@amadeuszsz Thanks
I changed to `ManagedTransformBuffer`
https://github.com/autowarefoundation/ManagedTransformBuffer