A True Zero-Copy RMW Implementation for ROS2

ROSCon 2019

Karsten Knese (BOSCH LLC)
karsten.knese@us.bosch.com

Michael Pöhnl (BOSCH GmbH)
michael.poehnl@de.bosch.com
A True Zero-Copy RMW Implementation for ROS2

The Use Case

automated driving is a data processing chain with a sensor input of up to 10 GB/s
A True Zero-Copy RMW Implementation for ROS2

The Problem

A typical middleware...

- copies when passing messages from the publisher to the middleware
- copies when passing messages from the middleware to the subscriber
- does internally even more copies and/or serialization/deserialization
- does at least n+1 copies for an inter-process-communication with n subscribers
A True Zero-Copy RMW Implementation for ROS2

The Problem

A typical middleware...
- copies when passing messages from the publisher to the middleware
- copies when passing messages from the middleware to the subscriber
- does internally even more copies and/or serialization/deserialization
- does at least n+1 copies for an inter-process-communication with n subscribers

No time to copy and serialize n GB/s while driving!
A True Zero-Copy RMW Implementation For ROS2
The Solution: True Zero-Copy Inter-Process-Communication

True zero-copy means...

- it is an end-to-end zero-copy approach from publishers to subscribers, based on shared memory
- the publisher directly writes to a chunk of memory provided by the middleware
- the middleware passes message references to subscribers and manages their liveliness

zero-copy communication is a must-have for automated driving!
A True Zero-Copy RMW Implementation for ROS2

The Solution: Eclipse iceoryx™

Eclipse iceoryx

- Shared memory inter-process-communication with zero-copy support
- Written in modern C++ with support for Linux and QNX
- Just launched as Eclipse incubation project with Apache 2.0 license

rmw_iceoryx – the iceoryx RMW implementation for ROS2

- First version available that supports publish/subscribe, the ROS2 CLI and a bridge
- Zero copy support for fixed size messages, slim serialization for dynamic messages
A True Zero-Copy RMW Implementation for ROS2

The Changes to ROS2 – Loaning Messages

1. borrow_loaned_message()
2. publish_loaned_message()
3. take_loaned_message()
4. return_loaned_message()
A True Zero-Copy RMW Implementation for ROS2

The Constraints for Zero-Copy

<table>
<thead>
<tr>
<th>dynamic_size_message.msg</th>
<th>fixed_size_message.msg</th>
</tr>
</thead>
<tbody>
<tr>
<td>int32 one_int</td>
<td>int32 one_int</td>
</tr>
<tr>
<td>float64 one_float</td>
<td>float64 one_float</td>
</tr>
<tr>
<td>char[] char_array</td>
<td>char[100] char_array</td>
</tr>
</tbody>
</table>

- [4 byte | 8 byte | 24 byte] dynamic size (heap allocation)
- [4 byte | 8 byte | 100 byte] fixed sized (POD)

The topic is not allowed to use heap-based data structures (e.g. STL containers with default allocators)
enough said ...
getting started

https://github.com/eclipse/iceoryx

https://github.com/ros2/rmw_iceoryx

https://github.com/karsten1987/fixed_size_ros2_demo