Introduction of a library for the replacement of heap allocation in Autoware #3274

sykwer started this conversation in **Design**

sykwer on Feb 16, 2023 Collaborator edited ▾

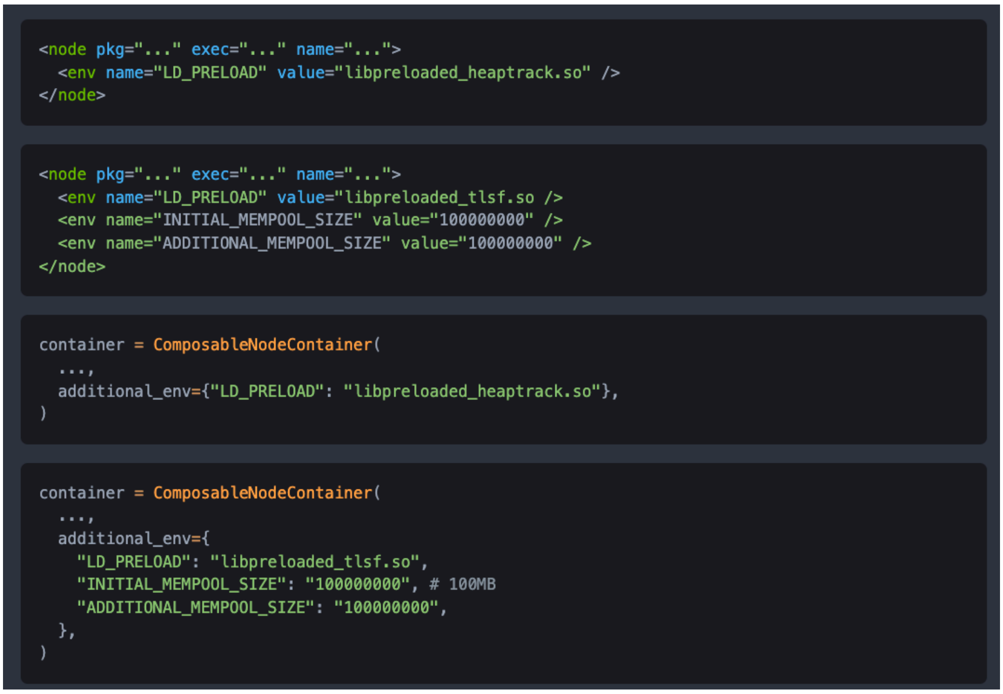
I've created a library named `heaphook` (https://github.com/tier4/heaphook), which replaces all the heap allocations in the specified ROS2 node. In this discussion thread, I'm going to explain the topics below and would like to obtain the community's consent to introduce this library.

- What is `heaphook`?
- Background and Performance Improvement
- What change is needed to Autoware repositories?
- User Story
- We need documentation
- Discussion

What is heaphook?

@sykwer developed a library that replaces all heap allocations (malloc, calloc, realloc, free, etc..) with an arbitrary memory allocator in the specified ROS2 node (more precisely, in the specified process).

You can apply this feature to any `Node` and `ComposableNodeContainer` by modifying a launch file as shown below.



```
<node pkg="..." exec="..." name="...">
  <env name="LD_PRELOAD" value="libpreloaded_heaptrack.so" />
</node>
```

```
<node pkg="..." exec="..." name="...">
  <env name="LD_PRELOAD" value="libpreloaded_tlsf.so />
  <env name="INITIAL_MEMPOOL_SIZE" value="100000000" />
  <env name="ADDITIONAL_MEMPOOL_SIZE" value="100000000" />
</node>
```

```
container = ComposableNodeContainer(
  ...,
  additional_env={"LD_PRELOAD": "libpreloaded_heaptrack.so"},
)
```

```
container = ComposableNodeContainer(
  ...,
  additional_env={
    "LD_PRELOAD": "libpreloaded_tlsf.so",
    "INITIAL_MEMPOOL_SIZE": "100000000", # 100MB
    "ADDITIONAL_MEMPOOL_SIZE": "100000000",
  },
)
```

As you can see, it's extremely easy to introduce the library and replace all the heap allocation functionality. Just specify the library in `LD_PRELOAD`.

Category

Design

Labels

type:new-feature

version:autoware-u...

3 participants

For now, two types of allocators are provided.

- `libpreloaded_heaptrack.so`: uses a standard malloc runtime and generates a log file to visualize heap usage over time, and is assumed to be used to capture the initial memory pool size required by the TLSF allocator.
- `libpreloaded_tlsf.so`: TLSF (Two Level Segregate Fit) allocator.

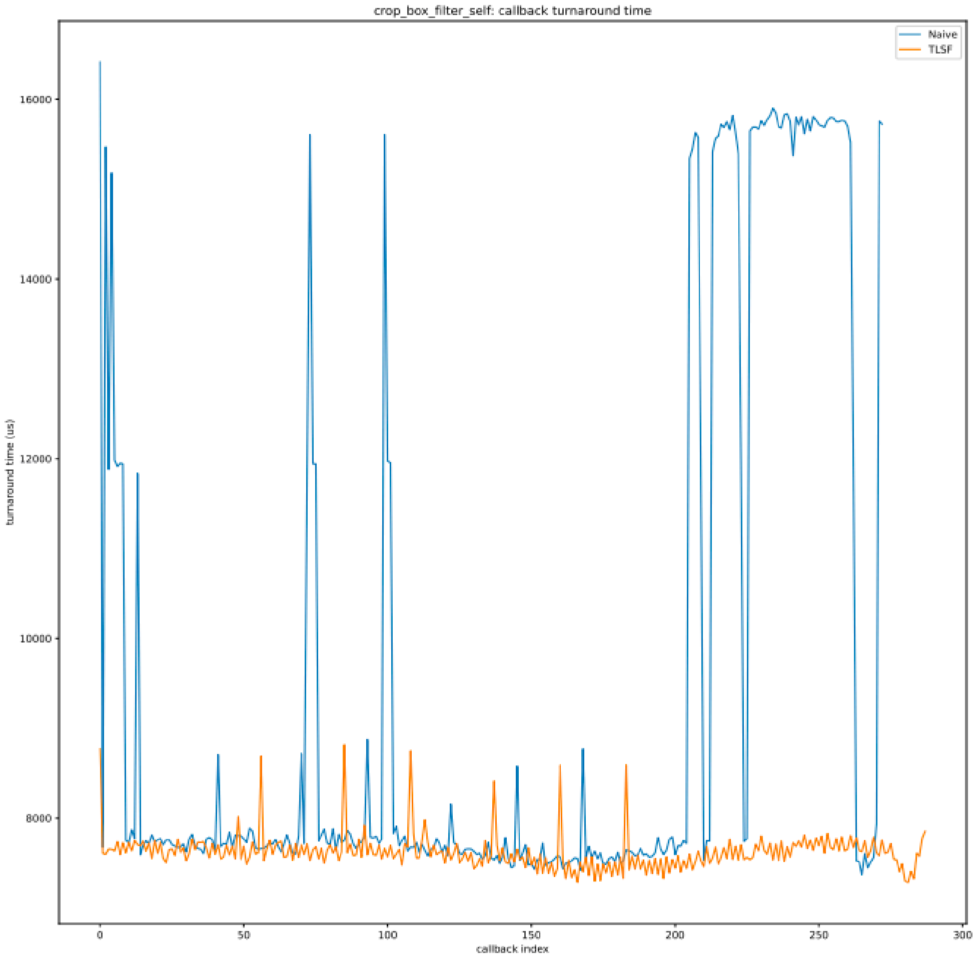
See the README page for more details.

Background and Performance Improvement

In the standard malloc runtime implementation, a large memory allocation request (e.g. malloc function call) triggers `mmap(2)` syscall. The first time the user program touches the mapped memory area, soft page faults occur, resulting in a large overhead. Therefore, in real-time systems, all virtual address space resources used for heap allocation should be allocated by `mmap(2)` at start-up and "first touch" in advance.

Additionally, there are recommended heap allocators for real-time systems and the default heap allocator should not be used. So heap allocators should be freely interchangeable by the Autoware users.

Our performance analysis shows that the response time bottleneck for each node of the `PointCloud Preprocessor` is due to the soft page faults. For example, the response time in `crop_box_filter_self` main callback can be improved as shown in below.



Measurement condition:

- Ubuntu22.04 + ROS2 Humble + Autoware Universe rosbag simulation
- Core Isolated
- Core Frequency Fixed (2.6GHz)
- L3 Cache: 12MB

What change is needed to Autoware repositories?

1. Add https://github.com/tier4/heap_hook to the `autoware.repos` file in https://github.com/autowarefoundation/autoware.
2. Description of dependent library settings.
3. Specify environment variables for the target node: `LD_PRELOAD=libpreloaded_tlsf.so`, `INITIAL_MEMPOOL_SIZE=...` and `ADDITIONAL_MEMPOOL_SIZE=...`.

For the first target node, I'm going to specify the pointcloud preprocessor container (here: https://github.com/autowarefoundation/sample_sensor_kit_launch/blob/6275063720f6fc94b5e2b477337b4246889c4c2a/common_sensor_launch/launch/velodyne_node_container.launch.py#L167-L175).
According to my measurement by `libpreloaded_heaphook.so`, `INITIAL_MEMPOOL_SIZE=10MB` is enough for the default rosbag simulation.

User Story

This library works as transparently as possible for Autoware users. Even if you don't know about `heaphook`, Autoware works fine in terms of logical output.
However, for high performance, the following steps need to be taken.

1. Grasp the maximum heap consumption

Set `libpreloaded_heaptrack.so` and run Autoware.

```
<node pkg="..." exec="..." name="...">
  <env name="LD_PRELOAD" value="libpreloaded_heaptrack.so" />
</node>
```

After running Autoware, you can get a log file under the current working directory in the format `heaplog.{pid}.log`.
You can visualize heap consumption transitions in PDF format based on the generated log file.

```
python3 heaplog_parser.py heaplog.{%pid}.log
```

2. Set TLSF Allocator with appropriate settings

Based on the information obtained in the step1, configure the TLSF allocator appropreately.

```
<node pkg="..." exec="..." name="...">
  <env name="LD_PRELOAD" value="libpreloaded_tlsf.so />
  <env name="INITIAL_MEMPOOL_SIZE" value="..." />
  <env name="ADDITIONAL_MEMPOOL_SIZE" value="..." />
</node>
```

We need documentation

The appropriate `INITIAL_MEMPOOL_SIZE` is highly dependent on the environment in which Autoware is running and is expected to be set by the Autoware operator themselves (The default configuration values should be those that work well with the rosbag in the Autoware Documentation). Therefore, we need to prepare appropriate documentation for Autoware users and encourage them to set appropriate values for the node with heavy heap consumption.

Discussion

- Where to manage the documentation
- Any design improvement
- Needs Tests?

↑ 2 👍 11 ❤️ 3 🚀 6

4 comments

Oldest Newest Top

sykwer on Mar 3, 2023 Collaborator Author edited ▾

Added to `autoware.repos` in #3306

↑ 1 👍 2 0 replies

sykwer on Mar 7, 2023 Collaborator Author

Roadmap to stable debian package: #3310

↑ 1 👍 2 0 replies

This comment was marked as off-topic. Sign in to view

xmfcx on May 10, 2023 Maintainer edited ▾

@sykwer thanks a lot for working on this.

I've merged #3306

> Where to manage the documentation

Could you create the steps to set up, run and adjust the parameters in the how-to-guides folder?

I think `configuring-heaphook.md` could be a good file name. Once it's created, we can relocate it if necessary.

> Needs Tests?

Once the documentation is out, we would like to test it on Leo Drive's vehicles to evaluate its performance.

For the unit tests, I think for this repo and the way it's integrated, integration tests (where multiple nodes run together in the test) could be suitable. But it's not too high priority just yet.

Do you have any specific nodes you'd like to run unit tests with it?

Launch file integration

Do you have plans to make it a part of autoware.universe launch files?

↑ 1

0 replies