19 October 2021

ROS 2 Executor: How to make it efficient, real-time and deterministic?

Callback-group-level Executor

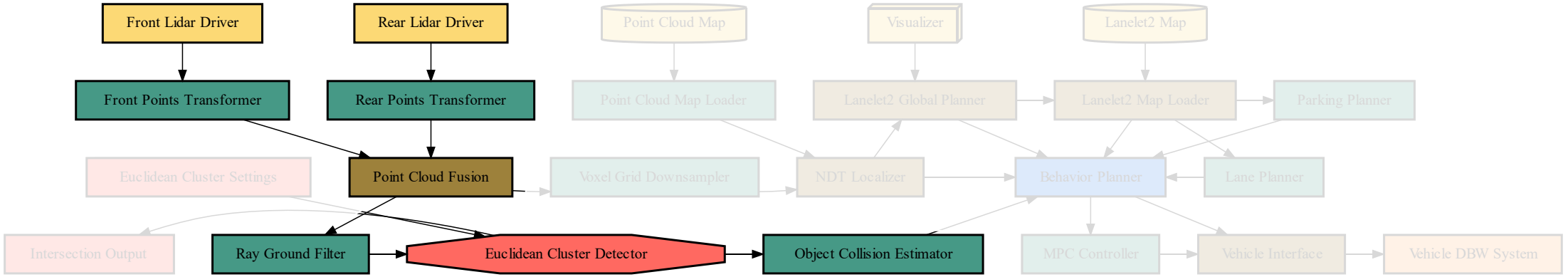
Dr. Ralph Lange
Bosch Corporate Research

BOSCH

Let’s prioritize the critical path …

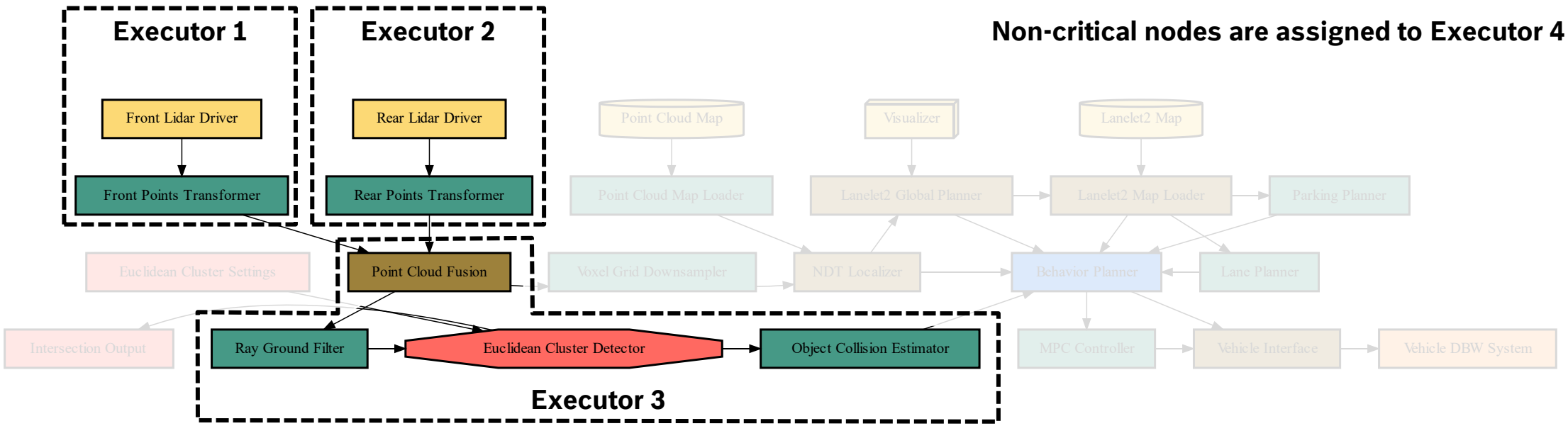
BOSCH

Critical Path in the Reference System



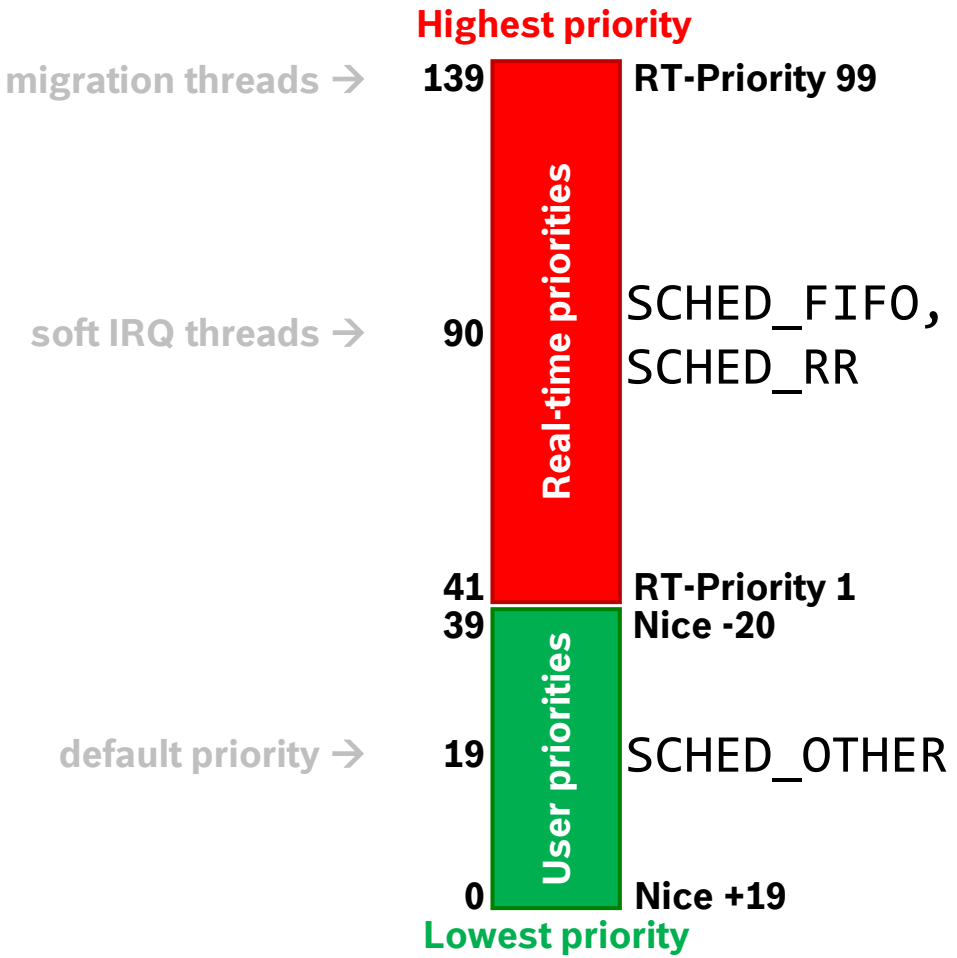
BOSCH

Distribution to Executors



BOSCH

Linux Schedulers and Priorities



SCHED_FIFO and SCHED_RR

- Defined in POSIX standard
- Preemption according to priority levels
- FIFO or round-robin (configurable quantums)

SCHED_OTHER

- Completely Fair Scheduler (CFS) by Ingo Molnár
- Since Linux 2.6.23
- Actually, three policies SCHED_NORMAL (aka SCHED_OTHER), SCHED_BATCH, and SCHED_IDLE

Real-time throttling:

- Non-RT schedulers get at least 50 ms per second to avoid that system hangs

Linux Schedulers and Priorities

```
USER         PID CMD                         PRI RTPRIO  NI
root           1 /sbin/init splash            19      -   0
root           2 [kthreadd]                   19      -   0
root           3 [rcu_gp]                     39      - -20
root           4 [rcu_par_gp]                 39      - -20
root           5 [kworker/0:0-events]         19      -   0
root           6 [kworker/0:0H-events_highpr  39      - -20
root           7 [kworker/0:1-events]         19      -   0
root           8 [kworker/u8:0-events_unboun  19      -   0
root           9 [mm_percpu_wq]               39      - -20
root          10 [rcu_tasks_rude_]            19      -   0
root          11 [rcu_tasks_trace]            19      -   0
root          12 [ksoftirqd/0]                19      -   0
root          13 [rcu_sched]                  19      -   0
root          14 [migration/0]               139     99   -
root          15 [idle_inject/0]              90     50   -
root          16 [cpuhp/0]                    19      -   0
root          17 [cpuhp/1]                    19      -   0
root          18 [idle_inject/1]              90     50   -
root          19 [migration/1]               139     99   -
```

ps ax --format uname,pid,cmd,pri,rtprio,nice

Prioritization of Executors

- Prioritization with nice levels in SCHED_OTHER

```
SingleThreadedExecutor executor1;
executor1.add_node(front_lidar_driver_node);
[...] // Add more nodes.

auto executor1_thread = std::thread(
  [&]() {
    int nice = -5;  // -20 to 19
    setpriority(PRIO_PROCESS, gettid(), nice);
    executor1.spin();
  });

[...] // Create other threads

executor1_thread.join();
```

- Real-time prioritization with SCHED_FIFO

```
SingleThreadedExecutor executor1;
executor1.add_node(front_lidar_driver_node);
[...] // Add more nodes.

auto executor1_thread = std::thread(
  [&]() {
    executor1.spin();
  });

auto handle1 = executor1_thread.native_handle();
sched_param params;  int policy;
pthread_getschedparam(handle1, &policy, &params);
params.sched_priority = 20;  // 1 to 99
pthread_setschedparam(handle1, SCHED_FIFO, &params);

[...] // Create other threads

executor1_thread.join();
```

Prioritization of Executors

- Prioritization with nice levels in SCHED_OTHER

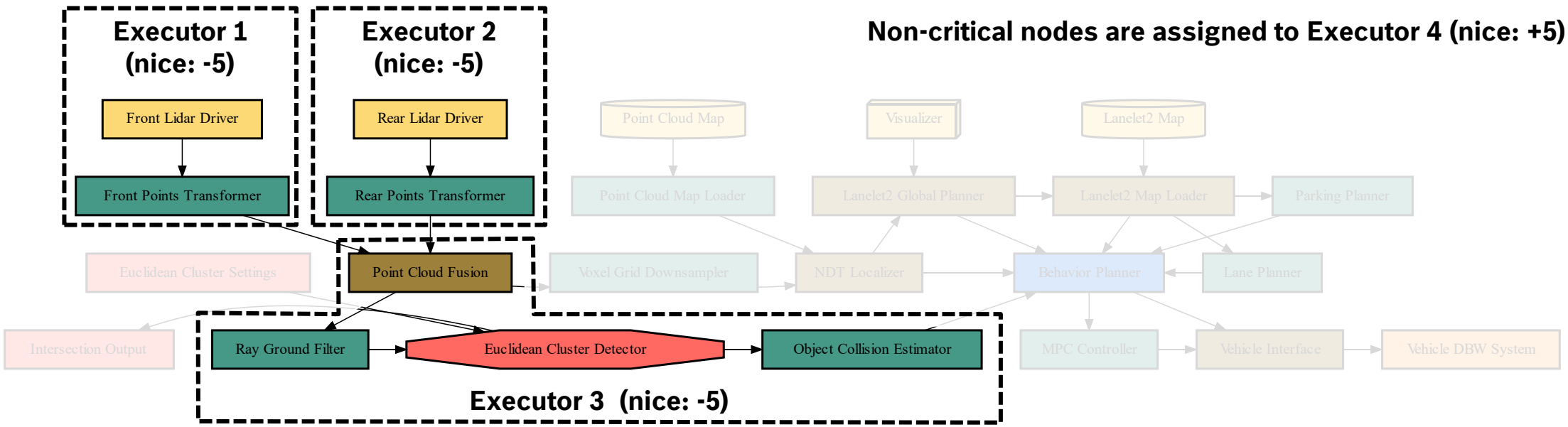
```
SingleThreadedExecutor executor1;
executor1.add_node(front_lidar_driver_node);
[...] // Add more nodes.

auto executor1_thread = std::thread(
  [&]() {
    int nice = -5;  // -20 to 19
    setpriority(PRIO_PROCESS, gettid(), nice);
    executor1.spin();
  });

[...] // Create other threads

executor1_thread.join();
```

- Real-time prioritization with SCHED_FIFO

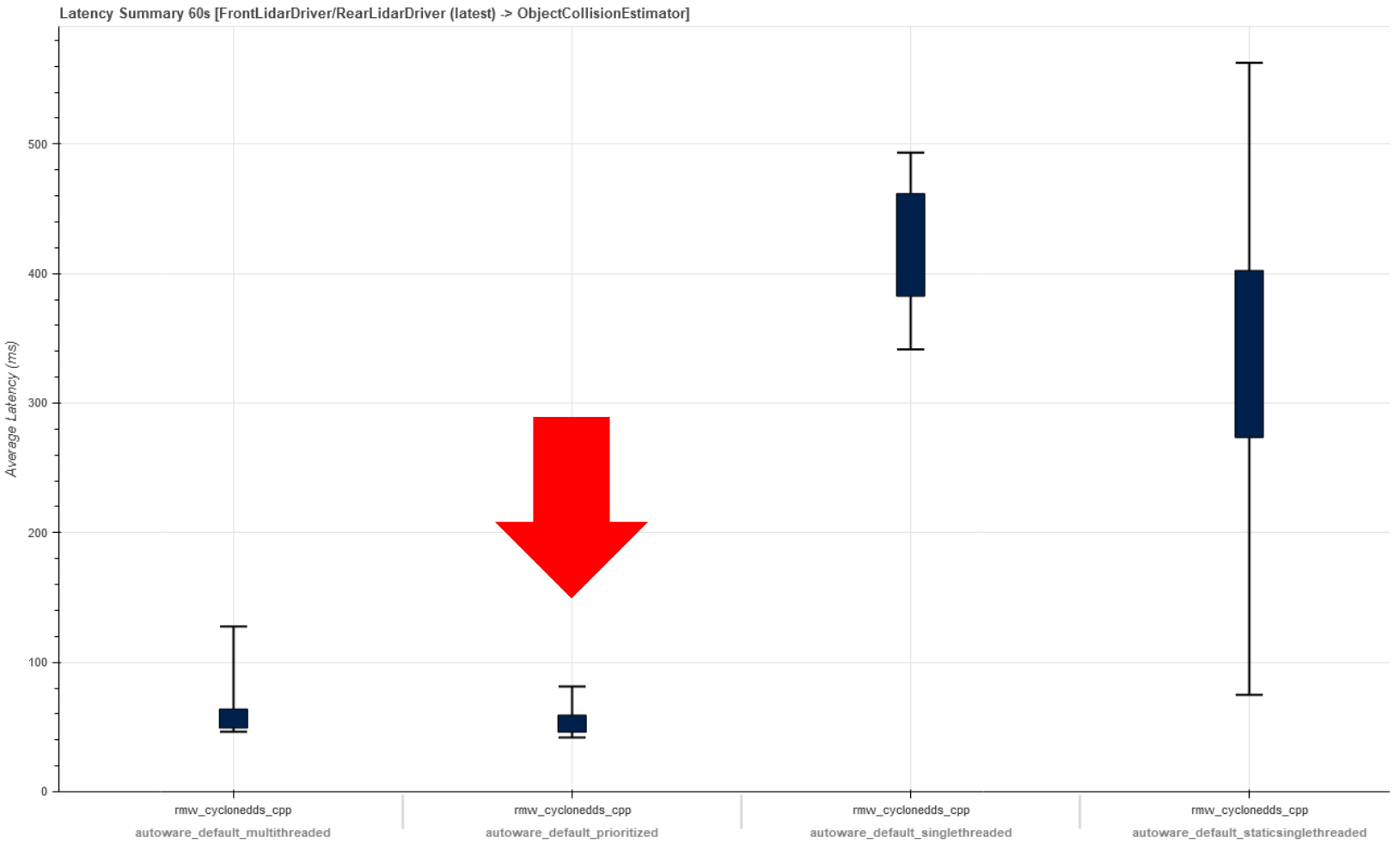
```
SingleThreadedExecutor executor1;
executor1.add_node(front_lidar_driver_node);
[...] // Add more nodes.

auto executor1_thread = std::thread(
  [&]() {
    executor1.spin();
  });

auto handle1 = executor1_thread.native_handle();
sched_param params;  int policy;
pthread_getschedparam(handle1, &policy, &params);
params.sched_priority = 20;  // 1 to 99
pthread_setschedparam(handle1,

[...] // Create other

executor1_thread.join()
```

- Not immediately possible for MultiThreadedExecutor
- See PiCAS talk by Hyunjong for a nice API

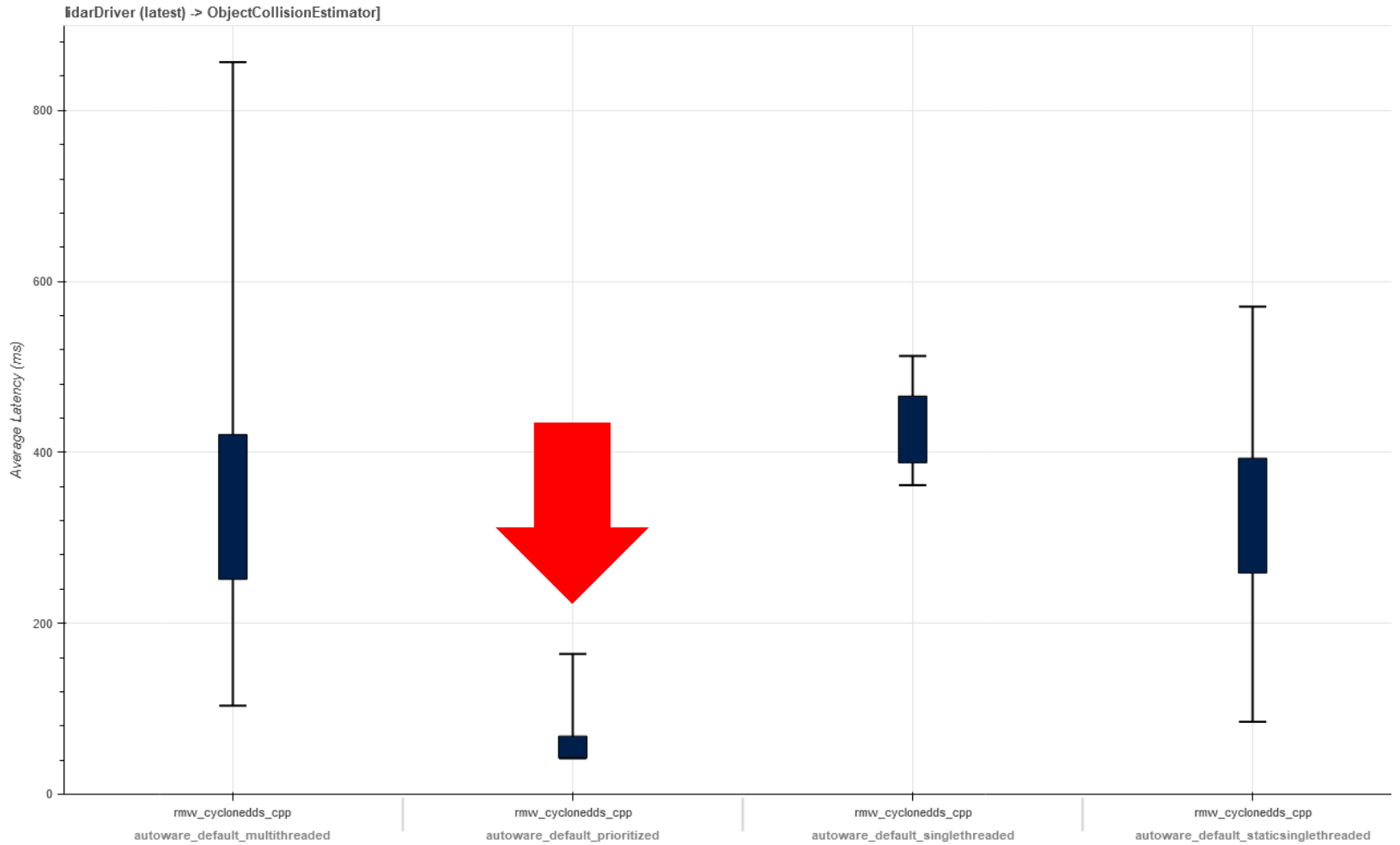
Distribution to Executors



Results for the Reference System (4 CPUs)



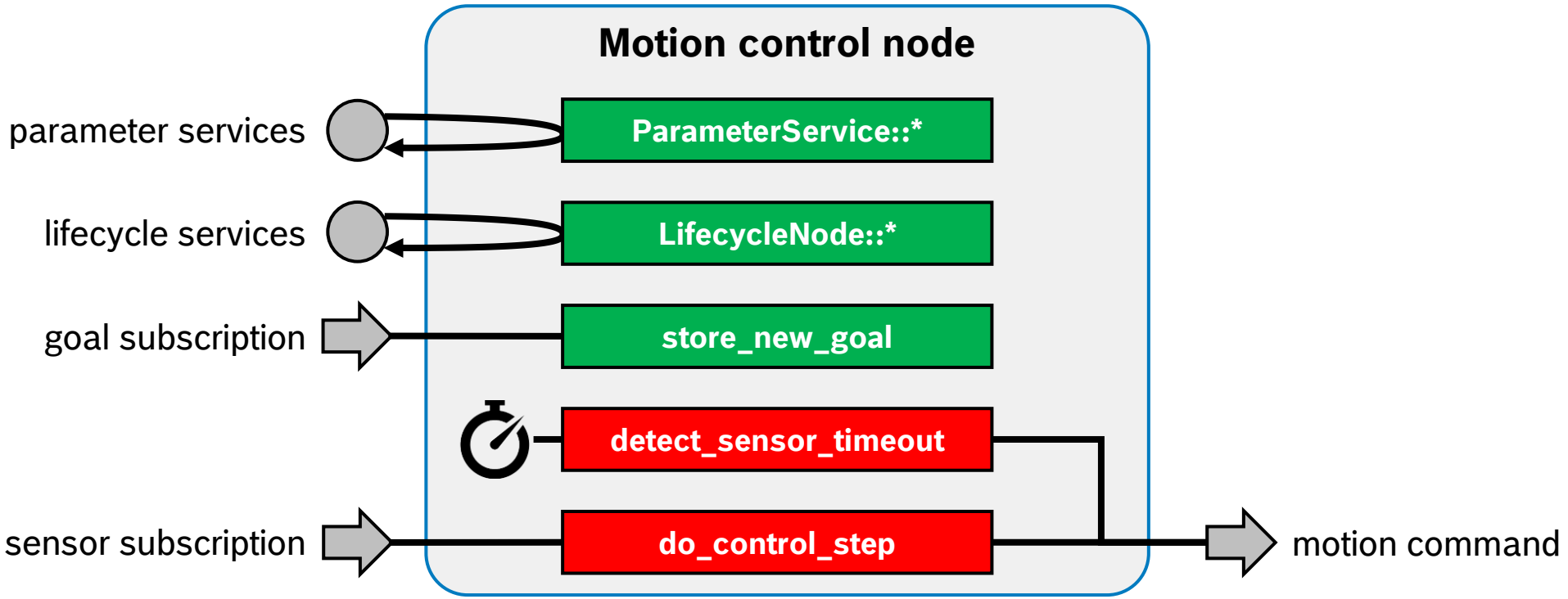
Results for the Reference System (2 CPUs)



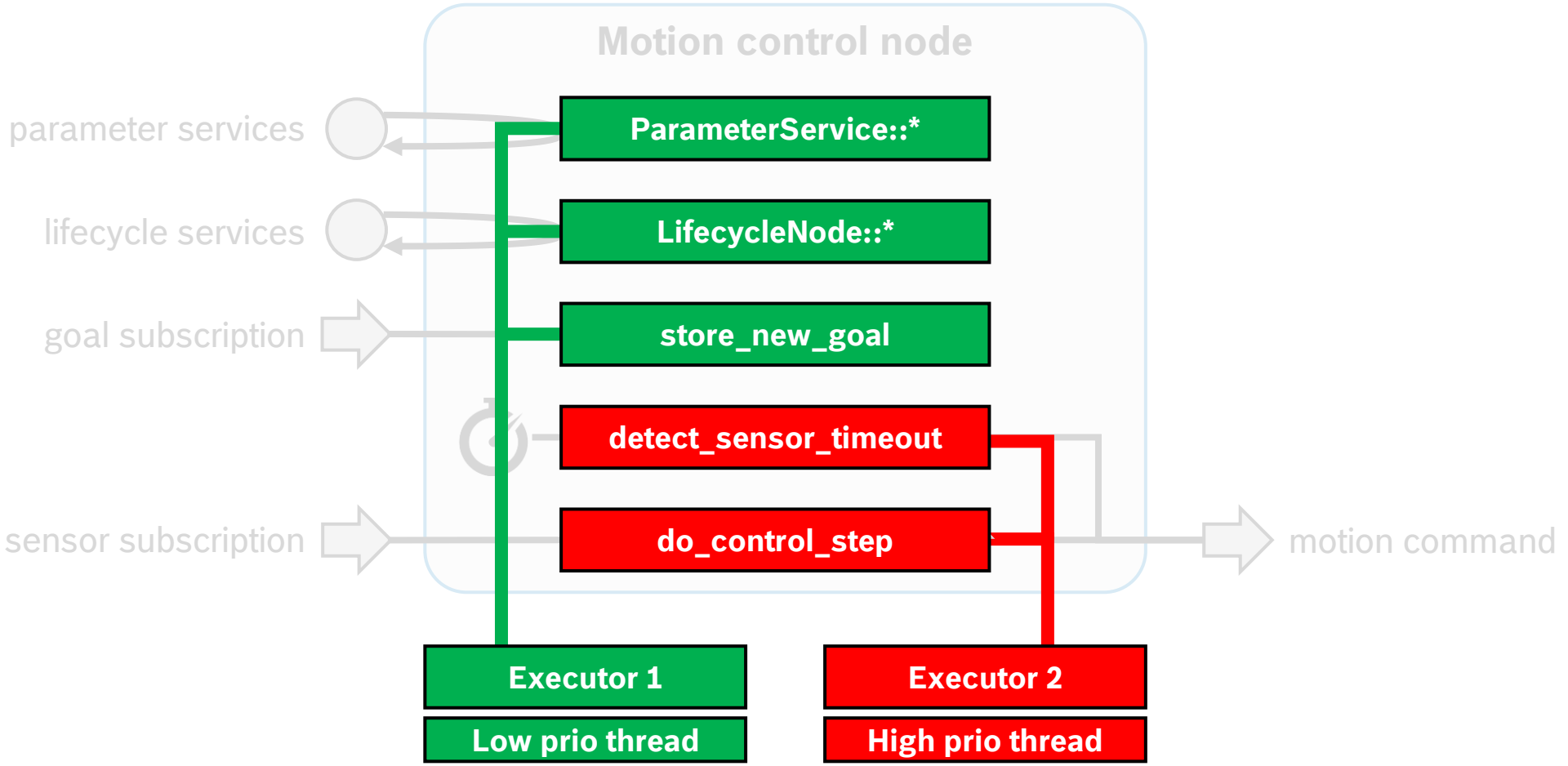
Callback-group-level Executor

Callback-group-level Executor is NOT another Executor

Motivation and Idea



Motivation and Idea



Executor API

base class Executor

- add_node
- remove_node

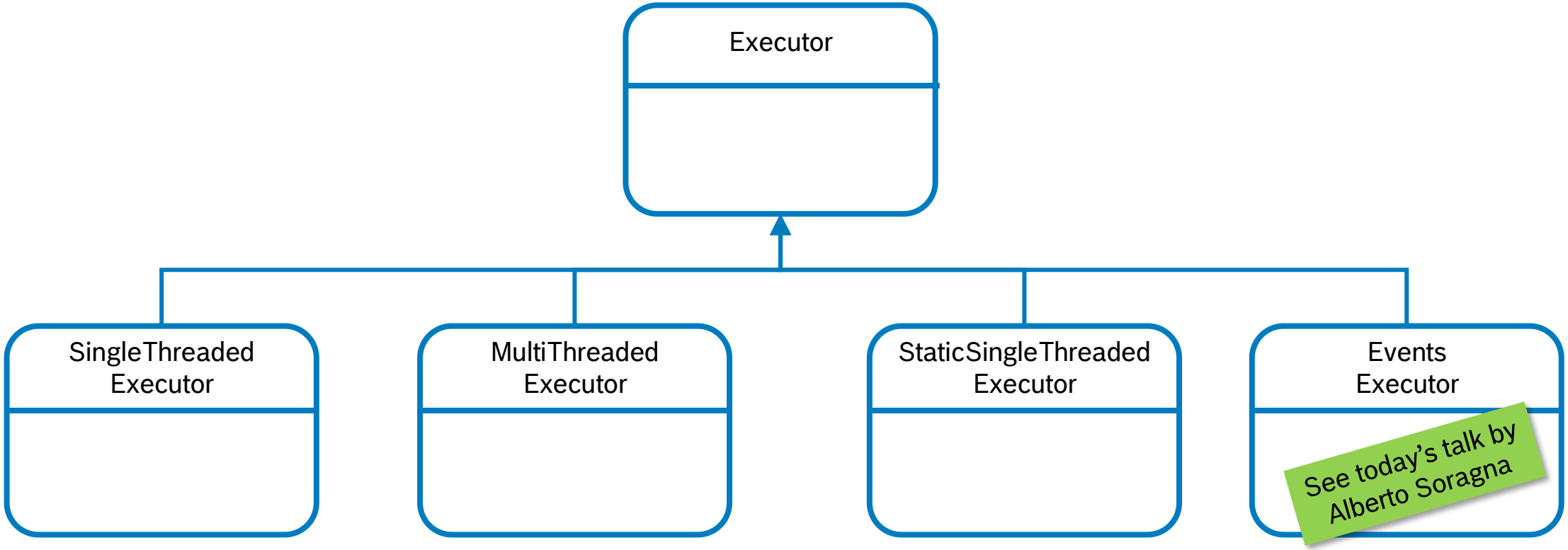
≤

base class Executor

- add_callback_group
- remove_callback_group
- add_node
- remove_node
- get_all_callback_groups
- get_manually_added_callback_groups
- get_automatically_added_callback_groups _from_nodes

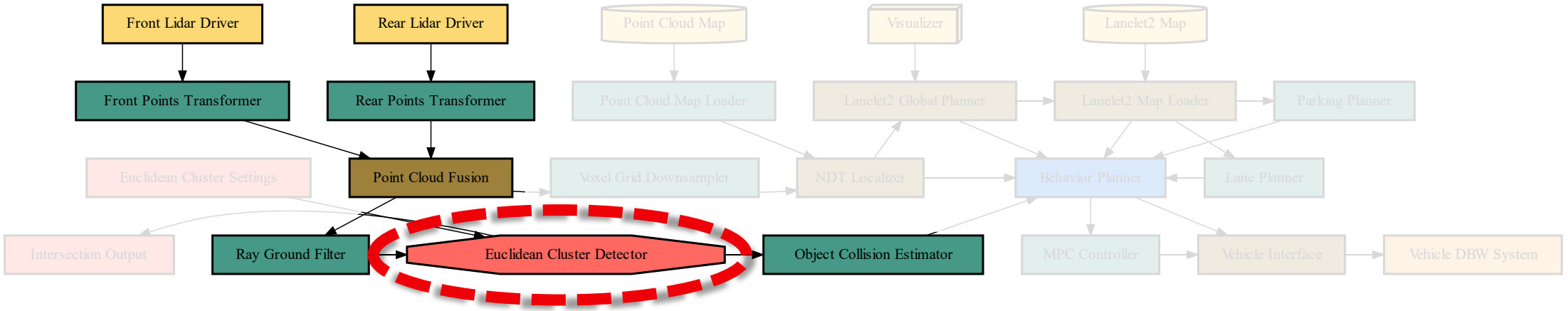
≥

Executor API (cont'd)

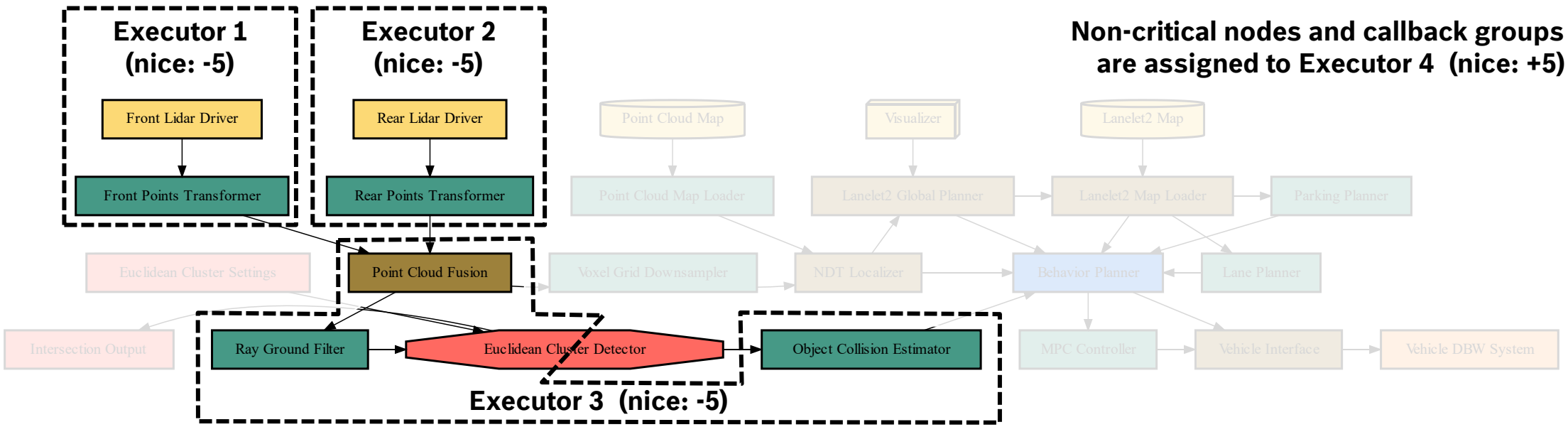


Many thanks to Pedro Pena (peterpena) and William Woodall (wjwwood) who brought the callback-group-executor prototype mainline!

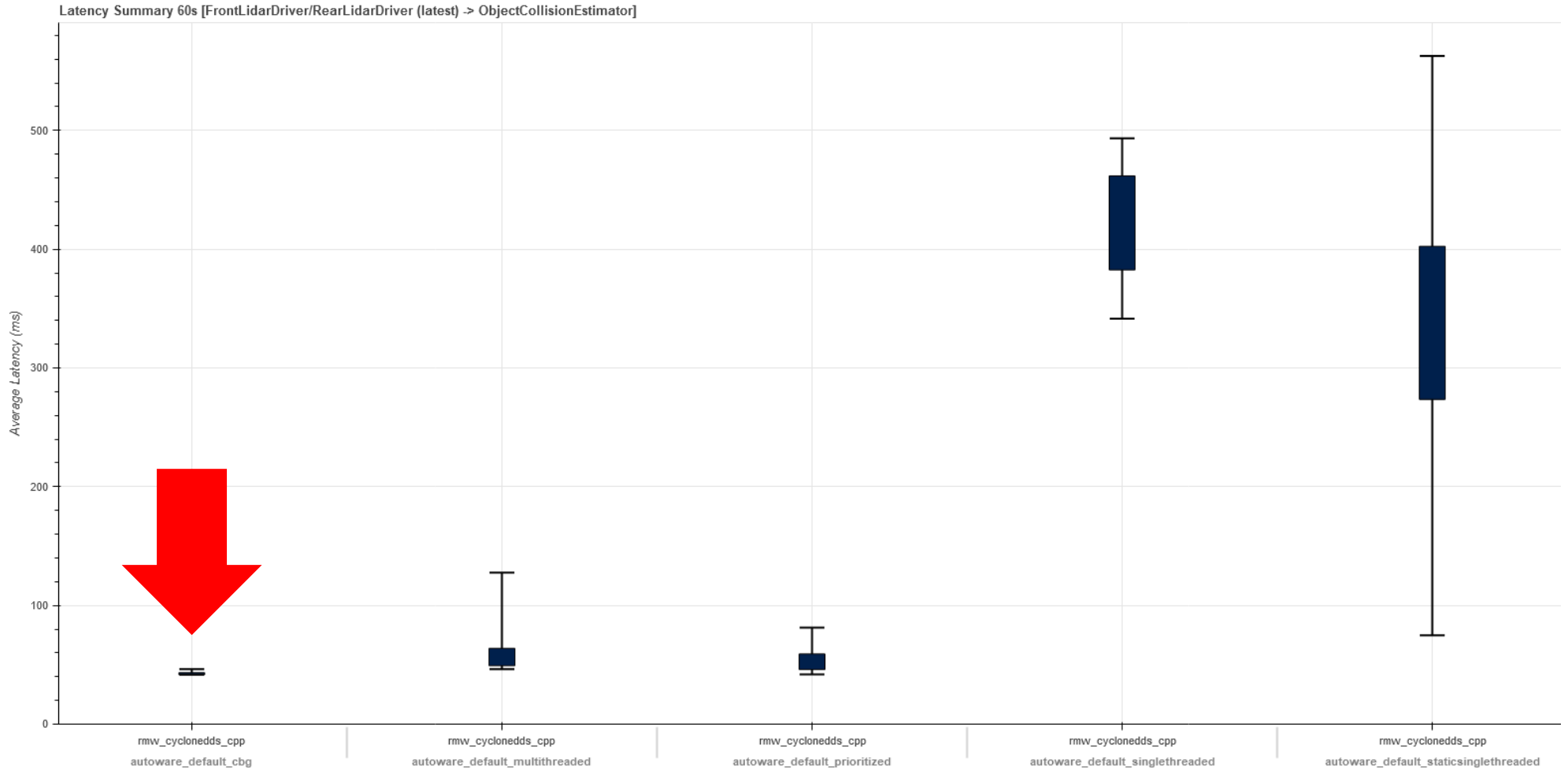
Critical Path in the Reference System



Distribution to Executors

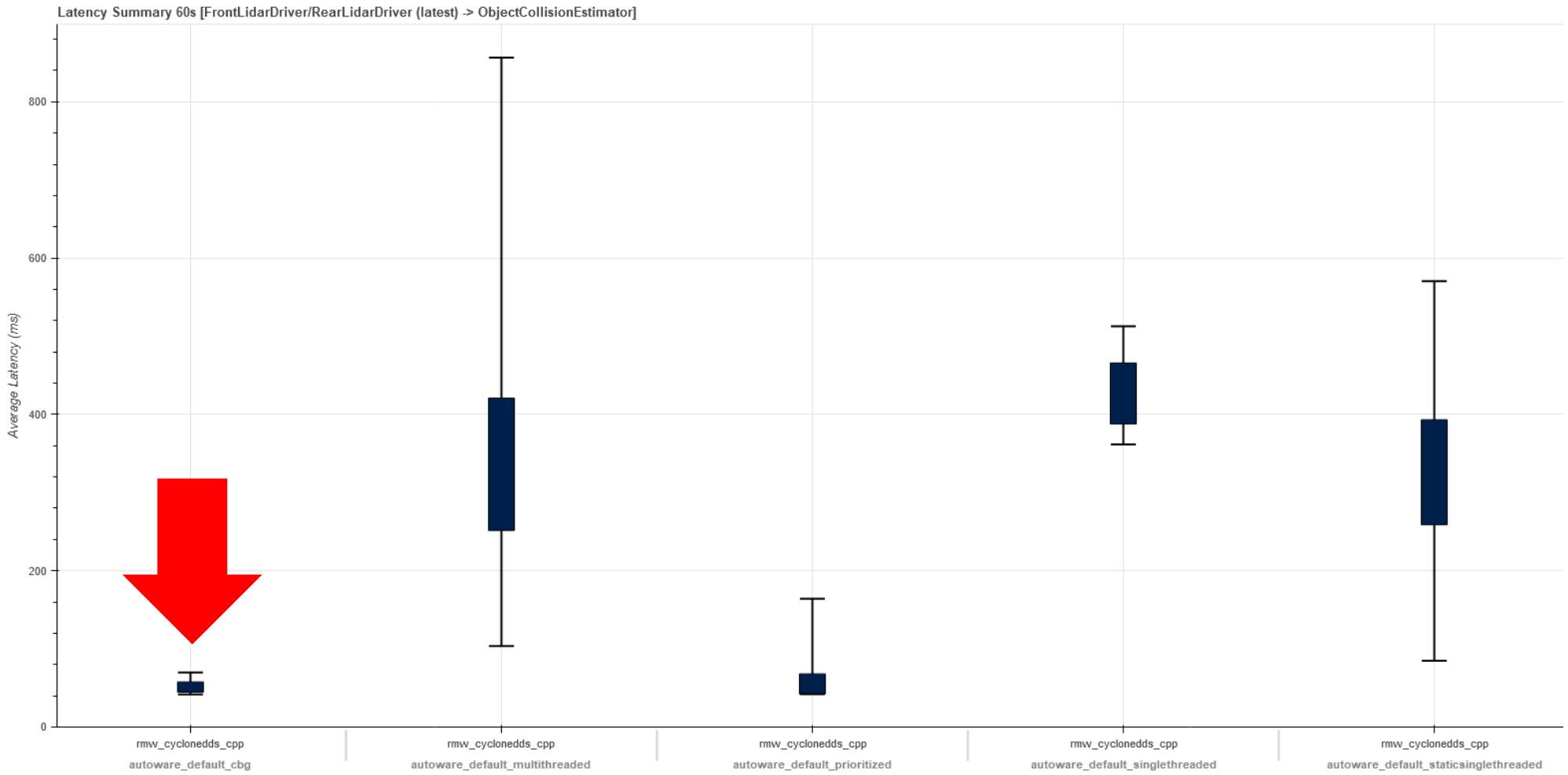


Results for the Reference System (4 CPUs)



BOSCH

Results for the Reference System (2 CPUs)

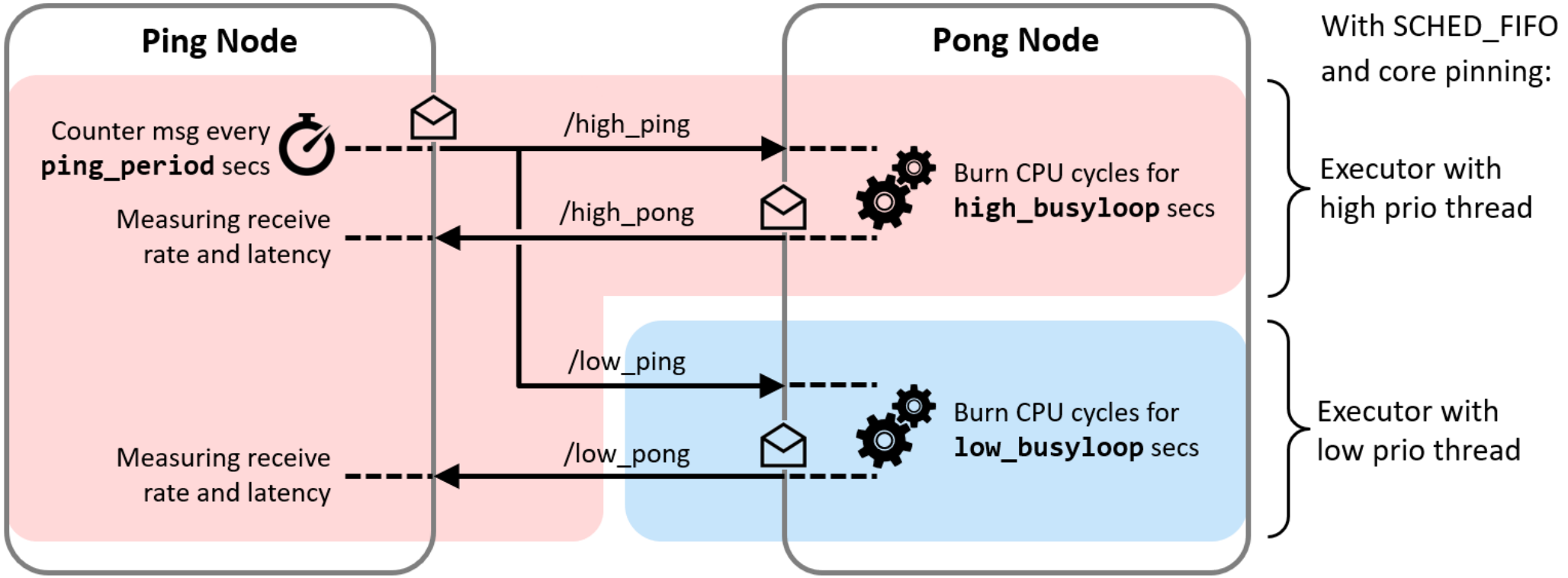


BOSCH

Demo Package

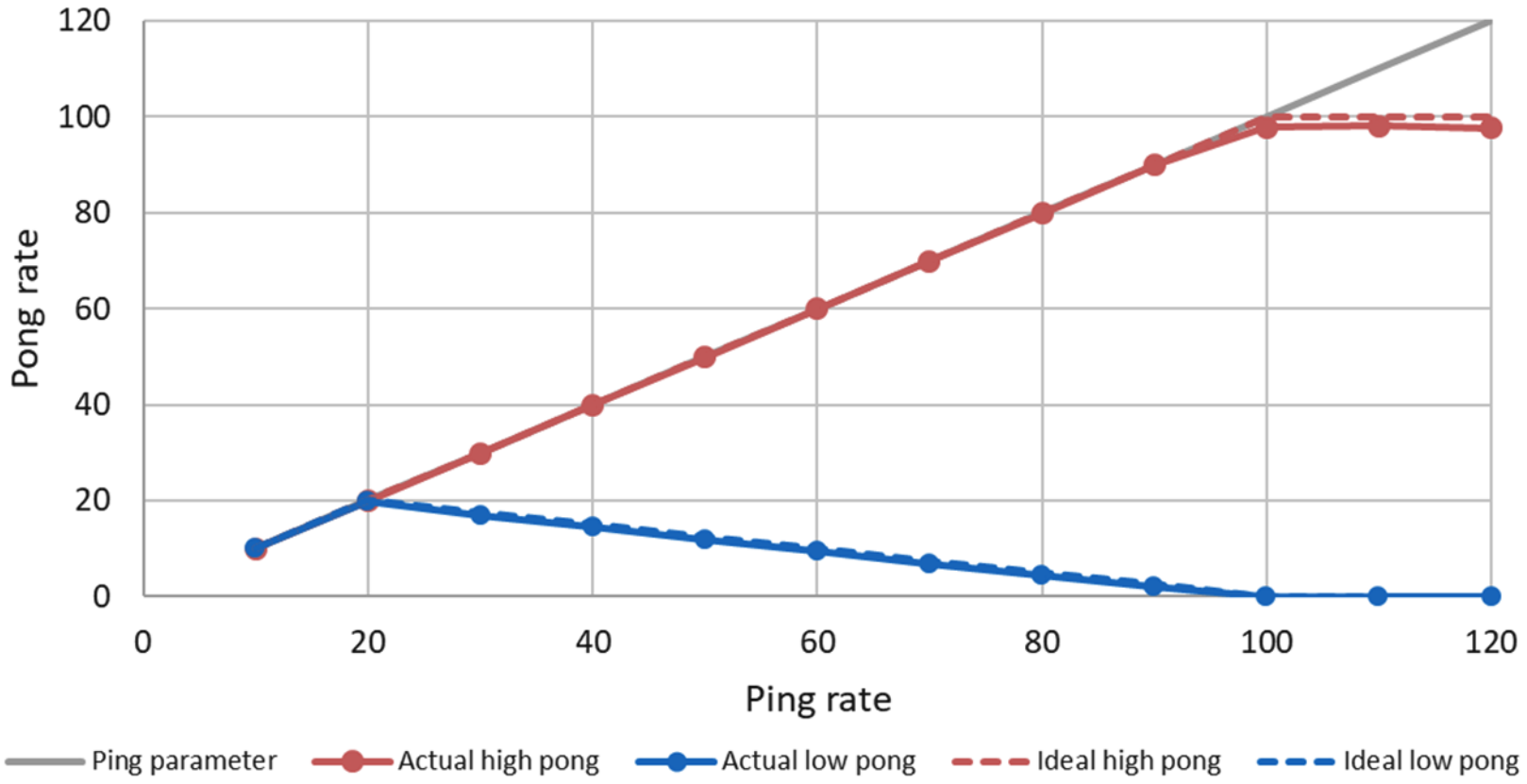
examples_rclcpp_cbg_executor

Package examples_rclcpp_cbg_executor



Source code at https://github.com/ros2/examples/

Package examples_rclcpp_cbg_executor




```
high_busyloop = 0.01 s

low_busyloop = 0.04 s
```

Source code at https://github.com/ros2/examples/

BOSCH

Looking forward to your questions!

Dr. Ralph Lange
Bosch Corporate Research

ralph.lange@de.bosch.com
github.com/ralph-lange

BOSCH