Implementation of a Fine-Grained Logging System for ROS2 Applications

GAI NAGAHASHI[†1] MASATO FUKUI[†2]
MIKIKO SATO[†1] TAKESHI OHKAWA[†1] MIDORI SUGAYA[†2]

Abstract: Currently, ROS is widely used as a framework for robot software development. When using ROS to realize complex cooperative behavior among robots, strict execution time logging such as node execution time and communication time is needed to analyze ROS nodes that communicate asynchronously. Although hCT (high precision Callback Tracer) has been proposed as one of the logging methods to obtain the execution time of ROS nodes, it is not supported in ROS2. This paper presents the porting of hCT performed on the Python platform in ROS2 and the performance measurement method that is planned to be developed in the future.

Keywords: ROS, execution time, measurement, performance evaluation

1. Introduction

In recent years, ROS (Robot Operating System) has become popular as a framework for robot software development. ROS is a set of software libraries and tools for building robot applications. Collecting data from a running system such as program execution time and communication time is necessary for debugging and performance improvement of robotic systems. To obtain node behavior information, logging systems and analysis tools[1][2] are needed.

Some logging systems and analysis tools available in ROS may support only ROS1 or ROS2. Allowing the same logging system and analysis tools to be used in ROS1 and ROS2 would be useful in terms of measuring and evaluating robot applications developed in ROS in the same environment.

The hCT (high-resolution Callback Tracer) [3] has been proposed as one of the logging methods to obtain ROS node execution time. hCT is a tool for ROS1 that can measure node execution time at a fine granularity without modifying the user application. However, hCT does not support ROS2. Therefore, in this study, we describe the design and implementation of ported hCT to ROS2 platform. Additionally, this paper introduces a performance measurement method of the robot system that is planned to be developed.

2. hCT in ROS2

2.1 Overview of hCT

hCT is implemented by extending roscpp, a C++ implementation of ROS1. hCT has the feature that logging can be performed without modifying the user application. Figure 1 shows the design and process flow of hCT. hCT acquires the time before and after calling the callback function (①, ②) and calculates the time difference (③). This is how the node execution time is calculated. In ROS, functions to be provided to nodes are generally implemented as callback functions. Therefore, callback execution time equals node execution time.

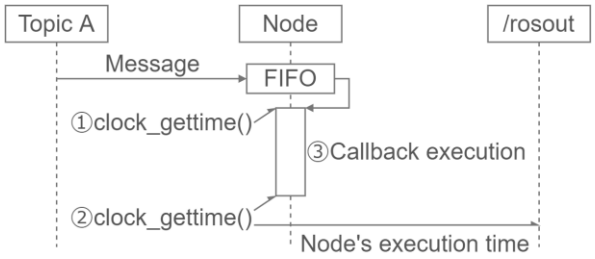

Figure 1 Design and process flow of hCT

2.2 Implementation of hCT in ROS2

hCT in ROS2 (referred to below as "hCT2") is an extended "rclpy" that is an implementation for Python in ROS2. We ported hCT2 in Python because Python was fewer porting steps than C++ and wanted to confirm that the hCT design can be implemented in Python. Figure 2 shows the design and process flow of hCT2. hCT2 is realized with only 7 lines of modification to rclpy in ROS2. Similar to hCT, hCT2 acquires the time before and after calling the callback function (①, ②) and calculates the time difference (③). The execution time of the node is output to the "/rosout" topic along with the callback function name (④). rosout is the name of the console log reporting mechanism in ROS, and "/rosout" topic is one of the components provided by rosout. Therefore, we implemented an "hct node" that saves the execution time of a node output to the "/rosout" topic to a file. The hct node gets node name that invoked the callback function, callback function name, and node execution time from the messages flowing to the "/rosout" topic. The acquired information is automatically saved in a csv file.

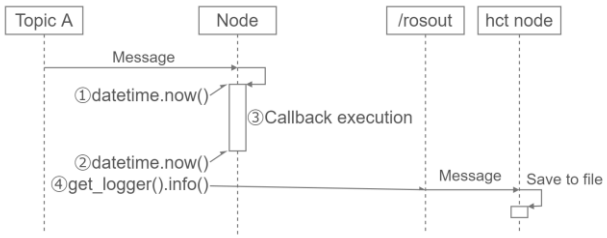

Figure 2 Design and process flow of hCT2

†1 Tokai University
†2 Shibaura Institute of Technology

3. Example of logging Pub/Sub communication

An example of hCT2 usage is shown using the py_pubsub program [4], which sends/receives strings using Pub/Sub communication. Figure 3 shows the connections of nodes and topics. Also, Table 1 shows the implementation environment.

Table 1 Porting and testing environment

Host OS	Windows 11 Home
CPU(Host OS)	11th Gen Intel(R) Core(TM) i7-1195G7
RAM(Host OS)	16GB
Virtualizer	Oracle VM Virtual Box (ver. 6.1.30)
Guest OS	Ubuntu 20.04.4 LTS
ROS2 distributions	Foxy Fitzroy

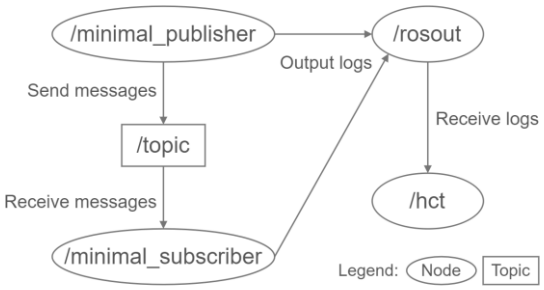

Figure 3 Connections of nodes and topics

The hct node gets node name that invoked the callback function, callback function name, and node execution time from the messages flowing to the "/rosout" topic (Figure 4). The hct node outputs the acquired information to the "/rosout" topic. For example, the center of Figure 4 shows that the listener_callback function is called by the minimal_subscriber node and the execution time of the node is 0.001838 seconds. Also, the hct node gets and saves acquired information from the messages flowing to the "/rosout" topic to csv file (Figure 5). For example, the third line of Figure 5 shows that the timer_callback function is called by the minimal_publisher node and the execution time of the node is 0.002041 seconds. Generation and saving of the csv file is done automatically by the hct node.

```
callback name:  timer_callback
execution time[sec]:  0.00468
-----
node name: minimal_subscriber
callback name:  listener_callback
execution time[sec]:  0.001838
-----
node name: minimal_subscriber
callback name:  listener_callback
execution time[sec]:  0.001307
```

Figure 4 Terminal screen of hct node

	A	B	C
1			
2	node name	callback name	execution time[sec]
3	minimal_publisher	timer_callback	0.002041
4	minimal_publisher	timer_callback	0.000898
5	minimal_subscriber	listener_callback	0.001434
6	minimal_subscriber	listener_callback	0.000519
7	minimal_publisher	timer_callback	0.002299

Figure 5 Node execution time recorded by hCT node(csv file)

4. Performance measurement methods planned for development

Performance measurement and evaluation using logging systems and analysis tools are necessary to confirm the operational stability of a robot system and how much heavy load processing is possible. Especially for large-scale systems or systems that require complex cooperative behavior among robots, it is important to guarantee the operational stability and to know the how much heavy load processing is possible.

Figure 6 shows the performance measurement method for the robot system. This proposed system measures and visualizes the robot system performance in real time and detect robot applications' error or bottleneck. For example, our proposed system can quickly resolve sudden failures by monitoring a robot system used at disaster sites.

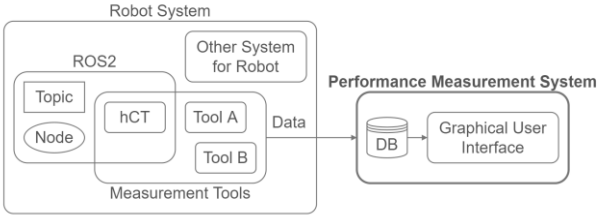

Figure 6 Measurement and visualization system when using hCT/hCT2 logging system

5. Conclusion

In this paper, we devised a method for porting the hCT (high-resolution Callback Tracer) proposed in ROS1, which can measure node execution time at a fine granularity, to ROS2 and implemented a prototype. Also, we introduced a performance measurement method of the robot system that is planned to be developed. In the future, we will be conducting detailed operational testing of hCT2 and defining requirements for a performance measurement method of the robot system that is planned to be developed.

Reference

[1] Christophe Bédard, "Tracing ROS 2 with ros2_tracing", ROSCon 2021, 2021-10-20
[2] "Chain-Aware ROS Evaluation Tool (CARET)", https://tier4.github.io/CARET_doc/latest/, (accessed 2022-09-17)
[3] Masato Fukui, Yoichi Ishiwata, Takeshi Ohkawa, Midori Sugaya, "Preliminary Implementation of Measurement Method for ROS1 Callback Execution Time", APRIS, 2022-01-28, http://id.nii.ac.jp/1001/00216093/, (accessed 2022-09-17)
[4] "Writing a simple publisher and subscriber (Python)", https://docs.ros.org/en/foxy/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Py-Publisher-And-Subscriber.html, (accessed 2022-09-17)