hCT : ROS1 でのコールバック関数実行時間計測ツールの設計と実装

福井誠人[1]　石綿陽一[2]　大川猛[3]　菅谷みどり[1]

概要：現在、自律移動ロボットのソフトウエア開発において、分散処理のフレームワークとして ROS（Robot Operating System）が広く使われている．ROS では、プロセスを node とし、node 間の通信を publish/ subscribe モデルで行う．これらの仕組みにより、ユーザーは複数のプロセスを持つ非同期システムを容易に構築できる．一方で、ROS では、subscriber がイベントにトリガーして行う処理(コールバック関数)の実行時間を測定することは難しい．そこで本研究では、ROS ミドルウェア上でコールバック関数が実行される箇所を特定し，それを計測用するためのツールを ROS1 上に設計・実装し，有効性を評価した．その結果，低オーバーヘッドでの調査に成功した．

キーワード：ROS，コールバック関数，実行時間，オーバーヘッド

1. はじめに

近年，ロボットソフトウェア開発における分散システムのフレームワークとして ROS(Robot Operating System)[1]が広く利用されている．ロボットソフトウェア開発は，ハードウエアを構成するセンサやアクチュエータを同期して動作させるため，ソフトウエア部品においても検知や制御に関わるデータ処理など様々なモジュールの開発が必要となる．ROS はこうした様々なモジュールをコンポーネントとして定義し，これらを比較的容易なプログラムのインターフェイスにより取り扱えるようにすることで，非同期分散型のソフトウエアを容易に構成できる利点がある．

具体的には，ROS はアプリケーションの機能を，異なる処理を実装した複数のコンポーネントとして分離して実装可能としている．また，この一つのコンポーネントを node として定義し，node 間の複雑なデータのやり取りや，通信を隠蔽して実装している．これにより，node として定義したコードの再利用性を高め生産性を向上させる．

ROS では node 間の通信は publisher/ subscriber（以降 Pub/Sub）モデルと呼ばれる方式で行う．publisher はデータを出版し，登録された subscriber がデータを受け取る．事前に登録された通信相手にデータを送信する仕組みは，ロボットなどのように特定の通信相手にデータを送る場合に適している．

publisher/ subscriber モデルの仕組みは，利便性が高い一方，複数の Pub/Sub 通信の混在によりアプリケーションの全体の構造は複雑化する．特に，複数の node ペアで Pub/Sub 通信を行い構成される大規模なアプリケーションでは，非同期に多数の node が動作するため，全体の性能を正しく評価することが困難である．例えば，あるアプリケーションの性能を評価しようとした場合，それを構成する全ての node 間の通信を把握する必要がある．しかし，Pub/Sub の通信は非同期で実行されており，この非同期時間を考慮した時間計測は現状十分に行われていない．加えて，アプリケーションの性能を把握するには通信時間だけでなくメッセージの処理にかかる時間も詳細に計測する必要がある．

そこで，本研究では，ROS アプリケーションの性能を正確に把握することを目的とし，ROS の特に callback 時の node の実行時間を計測するための仕組みを提案する．また，提案した仕組みを ROS に実装した．計測ツールのオーバーヘッドを評価するため Pub/ Sub 通信を行う 1 対の node ペアを作成し，計測ツールの有無でのアプリケーション全体の実行時間を測定した．その結果，設計したツールのオーバーヘッドを評価したところオーバーヘッドは十分に小さいことが分かった．

本論文の構成は次の通りである．2 節にて，ROS での計測の課題,3 節にて提案手法を説明する．4 節にて提案手法を使った時間計測, 5 節でまとめを述べる．

2. ROS アプリ計測の課題と提案

2.1 基本動作

本節では，ROS の基本構造について述べる.ROS ではソフトウエアの基本単位として node が用いられる．これは OS におけるプロセスと同等である．ROS での通信は topic と呼ばれる通信経路を介した Pub/Sub 通信モデルが用いられる.図 1 に，例を示した．Node にはデータを送信する publisher とそれを subscribe し，更新があったときに受信及びデータの処理を行う subscriber がある．/talker_node が送信を行う publisher であり，/listener_node が受信する subscriber である．この 2 つの node の通信は/chatter topic を介して行う．

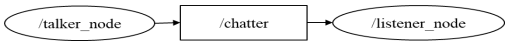

図 1 ROS publish/ subscriber モデル

始点となる publisher はセンサなどからデータを取得し，そのデータを他の node に渡す役目がある．データを受信した subscriber はデータの処理を行う．ROS には ROS1 と ROS2 の 2 種類あるが本研究では，現時点でより広く用いられている ROS1 を対象とした．

1 芝浦工業大学　2 株式会社 Ales　3 東海大学

2.2 ROS アプリケーションの実行時間計測の課題

node は入力となる topic を購読し，topic の更新をイベント待ちしている．事前にメッセージの受信をトリガーとしたハンドラ関数(コールバック関数) を node に設定する．メッセージを受信した時にコールバック関数が呼ばれデータの加工を行い，結果を別の名前の topic として出力する．この時の処理にかかる時間を node の実行時間と定義する．図 2 に ROS での通信のデータフローを示す．subscriber である node では topic1 から受信したメッセージをまず node の受信 queue に投入する．メッセージが FIFO queue に投入されたときに node に登録してあるハンドラ関数 (コールバック関数) がデータを取り出して処理を実行する．処理結果は別の topic を使い他の node に送信などがなされる．

node に登録してあるハンドラ関数 (コールバック関数) がデータを取り出して処理を実行するにあたり，メッセージが queue の中で待つ時間計測の実装はない．加えて queue があふれた場合は，メッセージは破棄される可能性もある．よって topic の監視のみでは node の実行時間を計測することは，困難である．このように ROS では一部の処理の実行時間を詳細に計測することが難しい課題がある．

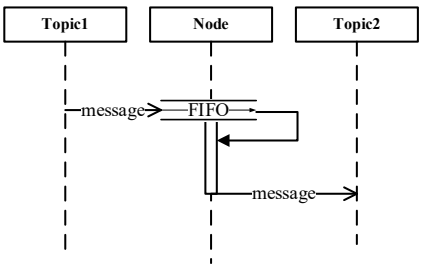

図 2 publish/ subscribe モデルでのデータフロー

これに対し，rosbag [2] は ROS の topic の記録及び再生するツールセットである．rosbag はメッセージ送信時間とメッセージの中身を記録することができる．rosbag では記録対象となる topic を subscribe し，topic の変更のたびにその時刻とデータの中身を記録する．しかし，rosbag では topic を対象としているため node の実行の計測はできない．

2.3 hCT: high precision Callback Trace

2.2 節に述べたように ROS の計測ツールでは，コールバック関数を対象とした計測ができない．そこで本研究では ROS node のコールバック関数をトレースするためのツールの設計，実装を行った．コールバック関数の呼び出し時にどの topic に紐づく関数が呼ばれたのかを特定するため，ROS 内部でコールバック関数をトラッキングする必要がある．ROS の内部では node，コールバック関数を特定するために 2 種類のキーが用いられる．node を生成しコールバック関数を設定する API では md5sum というハッシュ値が用いられる．これに対し，コールバック関数を実際に呼び出す API である CallbackQueue::CallOneCB() では removal_id という別のキーが用いられる．そこで，hCT ではコールバック関数を特定するためにこれら 2 つのキーをバインドして保存した．実行時間の計測にはコールバック関数の呼び出しの前後に clock_gettime()を用いて時刻を取得し，差分を計算した．この時刻の差分をもってコールバック関数実行時間とした．

3. 評価

提案した計測ツールを用いて ROS node の実行時間を計測したときのオーバーヘッドを評価した．実験は図 1 にある node 構成で/talker_node から/listener_node にメッセージを送信する．/listener_node ではメッセージ受信のたびにコールバック関数を実行する．この時 hCT を用いた実行時間の計測の有無により特定の回数のメッセージを受信するまでにかかる時間を time コマンドで計測した．実験では 100Hz でメッセージを送信し， 1 万回，5 万回，10 万回，50 万回受信するまでの 4 パターンで 4 回ずつ計測を行った．実験環境を表 1 に示す．ROS master 及び/talker_node, /listener_node は同一のホスト上で実行した．また，計測のたびにページキャッシュを削除して実験を行った．

CPU	Intel Xeon Gold 6230 (20Core, 2.1GHz, 27.5MB cache) x 2
RAM	1331GiB
OS	Ubuntu18.04.5 LTS
ROS	melodic

図 1 実験環境

実験結果を図 3 に示す．

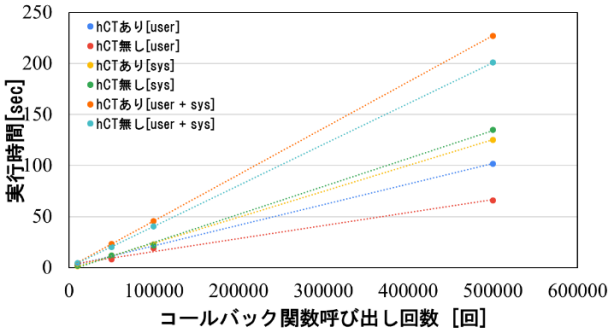

図 3 実験結果

結果から計測ツールのオーバーヘッドを計算する．コールバック関数 1 回あたりの計測ツールの実行時間は，およそ 0.054ms であった．

4. まとめ

本論文では ROS node のコールバック関数の実行時間計測方法について検討し，計測ツールのプロトタイプを実装した．評価ではオーバーヘッドが十分小さい事が分かった．

参考文献

1) M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, et al., "Ros: an open-source robot operating system", ICRA workshop on open source software, vol. 3, no. 3.2, pp. 5, 2009.
2) “rosbag - ROS Wiki”. https://wiki.ros.org/rosbag, (accessed 2021-10-18)