クラウドロボティクスにおける計算オフローディングに関する研究

2018SE009 古川湧也
指導教員：宮澤元

1 はじめに

近年, ロボットが様々な分野で利用されるようになり, ロボットの制御における演算処理が重要になっている. その一例として自律移動型のロボットが挙げられる. 自律移動型ロボットは IRobot のルンバや自動運転車などの複数の分野で用いられている. 自律移動を行うロボットに初めての環境でも障害物を回避しながら動作させる技術として Simultaneous Localization and Mapping(SLAM) がある. SLAM では, ロボットに搭載されたセンサを用いて測定した周囲の環境の情報を用いた演算処理によって, 自己位置の推定と環境地図の作成を同時に行う.

用途が多様化したロボット達を制御する技術としてクラウドロボティクスが注目されている [3]. これはロボットが行う処理が増えることによって, 大量の演算処理が要求される問題を解決するための技術である. クラウドロボティクスではロボットをインターネットに接続し演算処理をクラウドコンピューティング (クラウド) にオフロードする. これによってクラウドの計算リソースを用いて高度な演算処理が可能となる. 前述した SLAM の場合自己位置推定や地図作成の演算処理をクラウドで行うことでロボット本体の計算負荷を軽減できる.

クラウド上で行う演算処理は, 通常互いに隔離するように仮想化 (コンテナ化) されたアプリケーションとして扱われる. これらのコンテナがコンテナオーケストレータによってクラウド上の計算ノードにスケジューリングされることでアプリケーションが動作する.

クラウドロボティクスにおける演算処理のスケジューリングには、通常のクラウド上のコンテナスケジューリングとは異なった配慮が必要である. 例えばロボットを駆動させるようなリアルタイム性を要求する処理はロボット側で行うのが望ましい. 一方, クラウドにオフロードされる演算処理は必要な計算リソースを満たせるようにスケジュールされなければならない. そのためにコンテナの処理内容に応じた適切な実行場所を考慮してコンテナを配置する必要がある.

本研究の目的は, ロボットの演算性能や演算要求に応じて計算オフローディングを行うことができるようなコンテナオーケストレーションを実現することである。ロボット制御プログラムを要素ごとにコンテナとして構成し, 計算リソースに対するコンテナごとの要求とロボットの実行環境が持つ計算リソースとを勘案することで、コンテナを適切な実行場所に配置して計算オフローディングを行う.

本研究ではロボット制御プログラムをコンテナ化し, その動作確認を行った.

2 研究の背景

この節では, ROS と, コンテナオーケストレータ, およびクラウドロボティクスにおけるコンテナオーケストレータの課題について述べる.

2.1 ROS

Robot Operating System(ROS) とは Open Robotics で開発されたロボットのソフトウェアプラットフォームである [4].ROS にはプログラムのパッケージ管理, デバッグツール, 分散処理システム, ロボットハードウェアの抽象化といった特徴があり、ロボットに詳しくない者でも容易にロボットアプリケーションプログラムを開発できる [1].

ROS ではロボットを動作させるプログラムをノードという最小単位のプロセスに分割, コンポーネント化し、ノード間で pub/sub 通信を行いデータを送受信することでロボットを動作させている. これによって必要に応じてコンポーネントを組み替えることが可能になり, 柔軟性や再利用性の高いロボットプログラムを実現できる.

ROS のロボットプラットフォームに ROBOTIS の TurtleBot3[5] がある. これは LiDAR センサ,Raspberry Pi, ロボット制御ボード, モータを搭載した移動ロボットで ROBOTIS の e-manual によって手軽に ROS を用いたロボット操作を学習できる.

2.2 コンテナオーケストレータ

コンテナオーケストレータとはコンテナ仮想環境を管理運用するシステムである.

コンテナオーケストレータの代表例に Kubernetes[2] がある. Kubernetes ではポッドという 1 つ以上のコンテナを含んだ実行単位でコンテナを管理する. Kubernetes では作成されたポッドに計算ノードを割り当てることによって, ポッドが含むコンテナを実行する. またノードにラベル付けすることで Pod を配置する計算ノードを指定できる.

2.3 クラウドロボティクスにおけるコンテナオーケストレータの課題

クラウドロボティクスによってロボットは高性能な演算処理装置を搭載せずに演算処理機能を向上させることができる.

ロボットプログラムをコンテナ化しロボットはクラウドに計算負荷の高い処理をオフロードするなどの形で必要に応じて処理を分担することでクラウドロボティクスを実現できる. しかしコンテナオーケストレータが消費リソースだけを考慮してコンテナを配置してしまうとクラウド上で

ハードウェア制御などのロボット側で行うべき処理を行ってしまう恐れがある．さらに計算負荷の高い処理のオフロード先には適切な性能・計算リソースを備えた計算ノードを必要とする．他にもリアルタイム性の観点からオフロード先の通信レイテンシも考慮する．

このことからクラウドロボティクスではコンテナ毎に適切な実行場所を考慮したコンテナ配置を行う必要がある．

3 提案手法

本節ではコンテナオーケストレータを利用するクラウドロボティクスシステムを提案する．本システムでは 2 つのアプローチから提案手法を実現する．

はじめにロボット制御プログラムをコンテナ化を行う．ロボット制御プログラムを相互に通信し合う要素の集合として構成し，それぞれをコンテナとして動作させる．これにより，コンテナオーケストレータを用いてロボット制御プログラムをクラウドにデプロイして動作させることができる．

次にコンテナ化したロボット制御プログラムを要求リソースに基づいてスケジューリングする．通常のコンテナオーケストレータは，クラウドを構成する計算ノードの CPU およびメモリの空き容量に基づきスケジューリングを行っている．一方，ロボット制御プログラムはサーボモータやセンサ等の，通常のコンテナスケジューリングでは計算リソースとして考慮されないリソースを用いて動作する．そこで，コンテナスケジューリングに際してこのようなリソースを考慮するようにコンテナオーケストレータのリソース管理を拡張する．

4 提案手法の実現

提案手法を実現するために，ロボット制御プログラムのコンテナ化を試みた．表 1 に示す MPC をマスタノード，RPi4 と TB3 をワーカノードとする Kubernetes クラスタを構築した．ロボットの制御プログラムは ROS を用いて実現した．ROS プログラムはノードと呼ばれる最小単位のプロセスに分割して構成されており，ノードごとにコンテナとして実行することができる．

表 1 実験環境

名称	利用機器
MPC	PC(CPU: Intel Core i7-7700K, MEM:32GB)
RPi4	Raspberry Pi 4B(CPU:ARM Cortex-A72, MEM:4GB)
TB3	Raspberry Pi 3B(CPU:ARM Cortex-A53, MEM:1GB)

提案手法の内，ロボット制御プログラムのコンテナ化の実現性を確かめるために ROS プログラムをコンテナ化しその動作確認を行った．動作確認に用いたのは Teleoperation[5] という TurtleBot3 を遠隔 PC のキーボードからキー入力を行って操作する ROS プログラムである．Teleoperation を構成する ROS ノードを表 2 に示す．ROS の開発環境を持った Docker コンテナを複数作成しそれぞれに Teleoperation を構成するノードをダウンロードしてコンテナ化した．これらのコンテナを全て TB3 に配置し，以下の手順で動作することを確認した．

1. roscore コンテナを起動しマスタを立ち上げる．
2. bringup コンテナを起動し TB3 を制御可能な状態にする．
3. teleop コンテナを起動し TB3 に接続したキーボードからキー入力を行う．

動作確認の結果遠隔 PC を用いずにキー入力で TB3 を操作, すなわち Teleoperation を実行することができた．このことから ROS プログラムをコンテナ化することによってロボット制御プログラムのコンテナ化を実現できる．今回は TB3 に Kubernetes Pod として ROS コンテナを配置することはできず，Kubernetes クラスタにコンテナを分散配置した状態での動作確認は行っていない．

表 2 Teleoperation を構成する ROS ノード

名称	概要
roscore	ROS ノード同士が通信するためのマスタとして働く
teleop	遠隔 PC で動作してキー入力を受け取る
bringup	TurtleBot3 の制御を行う

5 おわりに

本研究では、ロボットの演算性能やリソース要求に応じて計算オフローディングを行うことができるようなコンテナオーケストレータを提案した．提案手法の実現性を確認するために，ROS プログラムのコンテナ化を行った．コンテナ化された ROS プログラムが実際にロボットの制御プログラムとして動作することを確認した．しかし ROS コンテナを Kubernetes クラスタに分散配置し動作させることができなかった．今後は作成した ROS コンテナを Kubernetes クラスタに分散配置し，提案手法に基づく計算オフローディングを可能とするようなコンテナオーケストレータを実現する．

参考文献

[1] 表允・倉爪亮・鄭黎: 『ROS ロボットプログラミングバイブル』．オーム社，東京，2018.

[2] 青山真也: 『Kubernetes 完全ガイド 第 2 版』．インプレス,2020.

[3] R. Chaari, et al., “ Towards a Distributed Computation Offloading Architecture for Cloud Robotics, “ in Proceedings of 15th International Wireless Communications & Mobile Computing Conference(IWCMC), 2019.

[4] ROS, https://www.ros.org (2021 年 9 月 30 日アクセス)

[5] TurtleBot3, https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/ (2022 年 1 月 18 日アクセス)