ロボット制御システムにおけるOSS機能モジュール向けサンドボックスの実現

b1017197 瀧本恒平
指導教員：松原克弥

Implementation of a Sandbox for an OSS Function Module in Robot Control Systems

Kouhei Takimoto

概要： ROSにおいて，OSSノードを組み合わせることによるロボットアプリケーションの開発が一般化している．これは，ROSの大規模なコミュニティにおいて大量のOSSノードが提供されているためである．しかし，動作の詳細を把握せずに組み込むことの多いOSSノードがシステム内に混在することで，組み込みシステムの限られた計算資源（CPU, メモリ, ネットワーク帯域等）の消費量に関する見積もりが困難になる．本研究では，OSSノードによるシステムへの想定外の負荷を防ぐことを目的として，ノードを対象としたサンドボックスの実現を提案する．本研究の実現により，ROSを用いた開発におけるOSSノードの活用を促すと共に，とロボット制御システムの安定性向上を可能とする．

キーワード：ROS, OSS, サンドボックス

Abstract: In ROS, the development of robot applications by combining OSS nodes is becoming more and more common. This is due to the large number of OSS nodes provided by the large ROS community. However, because OSS nodes are often installed in a system without knowing the details of their operation, it is difficult to estimate the consumption of limited computing resources（CPU, memory, network bandwidth, etc.）of an embedded system. In this research, we propose a node sandbox to prevent unexpected load on the system by OSS nodes. The realization of this research promotes the use of OSS nodes in the development of ROS and makes it possible to improve the stability of robot control systems.

Keywords: ROS, OSS, sandbox

1 背景

近年，様々な分野においてロボットの活用が拡がっている [1][2]．このロボットのシステム開発において，Robot Operating System（ROS）を用いる機会が増えている．ROSとは，ロボット開発を効率化するアプリケーションフレームワークのことであり，ノードと呼ばれるロボットの機能を実現するパッケージを複数組み合わせることでシステムを構築する．このROSのコミュニティにおいて，様々なロボットに必要な共通の機能（画像処理 [3] など）は，オープンソースソフトウェア（OSS）のノードとして提供されている．そのため，フルスクラッチで機能を実装していた従来のロボット開発に比べて，OSSノードを用いた開発は高速であることから，ROSは現在のロボット開発において重要なフレームワークのひとつとなっている．ROSには，ROS 1とROS 2の二種類が存在している．先にリリースされたROS 1では，ROS独自の実装によりノード間通信を可能としていたが，マスターと呼ばれる通信相手の問い合わせを行う仕組みが単一障害点となっていた．また，サポートしているOSもUbuntuのみであるなどの制約も存在した．ROS 2では，ROS 1で抱えていた問題点や出来なかったことを可能にするために，通信プロトコル等アーキテクチャが大きく変更されている．また，ブリッジというROS 2のパッケージを用いることで，開発途中のROS 2においてROS 1のノードを使用できるようになるため，今後はROS 2の利用拡大が予想される．そのため，本研究ではROS 2を対象とする．

2 課題

ROSでは，OSSノードの動作がシステム内の他のノードの動作に影響を与える可能性がある．OSSノードのような第三者が作成したノードは動作の詳細を把握せずにシステムに組み込むこ

とが多いためである．図 1 は，実際に OSS ノードを取り入れたロボット制御システム（ドローン）のイメージであり，フルスクラッチで実装した移動制御ノード，位置情報取得ノードに加え，OSS である画像処理ノードを取り入れたシステムの一例である．ここでは，OSS 画像処理ノードがシステム内の計算資源を専有してしまい，その影響を受けた移動制御ノードは必要分の計算資源を与えられていない．そのため，ロボット制御システムの動作が不安定になっている．

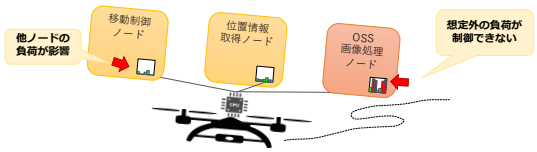

Fig. 1: ロボット制御システムの例

3　提案する手法

前章までで述べたとおり，ROS を用いたロボット制御システムの開発において OSS ノードを用いることには，同一システム内の他ノードの動作に影響を与える可能性があるという課題がある．本研究では，OSS ノードによるシステムへの想定外の負荷を防ぐことを目的として，システム内の各ノードを対象としたサンドボックスを作成し，計算資源消費量を実行時に制御することを提案する．ここでのサンドボックスとは，計算資源の最大消費可能量に制限をかける機構のことを指す．

3.1　サンドボックス設定のフロー

まず，サンドボックスの設定前に，シミュレータを用いて各ノードが使用する計算資源量の見積もりを行う．次に，見積もり結果から，サンドボックスで制限する計算資源量の上限を設定し，各ノードに対応したサンドボックスを作成する．これにより，図 2 のように OSS 画像処理ノードの使用可能な計算資源量の上限がコントロールされ，移動制御ノードは必要とする分の計算資源を使用できる．そのため，不安定であったロボット制御システムの動作が安定するようになる．

3.2　ノードの計算資源消費量見積もり手法

まず，ロボットを実環境で動作させる前に，各ノードが消費するおおよその計算資源量を見積もる必要がある．そのため，Gazebo**Gazebo** によってロボットを仮想環境で動作させる．Gazebo とは，ROS との連携が充実したオープンソースの 3D ロボットシミュレータである．これにより，各ノードを擬似的に動作させ，その動作中の各計算資源量を一定間隔で計測し，計算資源量を見積もる．この計算資源消費量の計測には，Linux の procfs と，帯域幅監視ツールの一つである NetHogs を使用する．procfs とは，プロセス情報の擬似ファイルシステムであり，ここでは CPU とメモリの使用率の取得に使用する．また，NetHogs は，プロセスごとのトラフィック量を取得するものであり，ここではネットワーク帯域幅の取得に使用する．これらの取得したデータをもとに見積もりを行う．

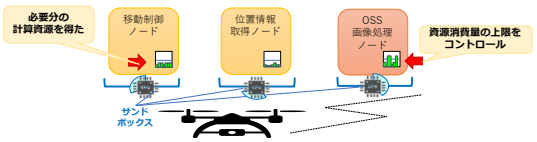

Fig. 2: サンドボックス導入後のロボット制御システムの例

3.3　サンドボックスの作成

本研究におけるサンドボックスとは，cgroup と tc コマンドを用いて，ノードごとの計算資源における最大消費量に対して制限を課す機構である．cgroup とは，ROS の動作プラットフォームである Linux 上で動作するコンテナ型仮想化技術の一つであり，グループ化されたプロセスの計算資源の利用を制限する Linux カーネルの機能である．また，tc コマンドは，ネットワークインターフェースに対してネットワーク帯域制限を設定する機能である．cgroup については，cgroup v1 の改良版である cgroup v2 を主に使用する．しかし，cgroup v2 は比較的新しい機能であるため，v1 で使用されていた機能全てを使えるわけではない．そのため，cgroup v1 と v2 を共存させる形で使用する．cgroup v1 では設定した計算資源量の上限を超過するとプロセスが停止してしまうが，cgroup v2 であればプロセスを停止させずに計算資源消費量を抑えることができる．cgroup の操作には，基本的に cgroupfs という擬似ファイルシステムを用いる．新たに cgroup を作成する際も，通常のファイルシステムを扱うように mkdir コマンドを用いることが可能である．

また, 制限を行う際には, cgroup のサブシステムを用いる．サブシステムとは，linux のプロセスに作用する計算資源制御機能のことを指す．具体的に cgroup を用いてノードの計算資源消費量の制限を行うには，まず cgroup を作成し，ノードの PID を cgroup.procs に登録する．その後，計算資源消費量の上限値を cpu.max 等のサブシステムに書き込むことで，CPU 使用率とメモリ使用率の制限を完了できる．また，ネットワーク帯域幅の制限についても，cgroup の net_cls サブシステムを用いて，パラメータとネットワークインタフェースを設定することで，制限を完了できる．これらの設定を行うことで，本研究におけるサンドボックスを作成する．

4 関連技術

本研究の関連技術として, 一般にコンテナ型仮想化技術と呼ばれる, アプリケーションコンテナがある．このアプリケーションコンテナの一例として，Docker**Docker** が存在する．Docker は，Linux カーネルの機能である cgroup と namespace を用いて計算資源の制限と分離が可能であることに加え, コンテナの実行に必要なパッケージの共有が容易であるなど, 多くの機能を持つ．そのため，Docker は計算資源の制御に特化している cgroup と比べてオーバーヘッドがかかる．本研究では, コンテナ型仮想化機構の一つである cgroup と linux の tc コマンドを用いて，ノードの計算資源消費量の管理に特化したサンドボックスを作成する．Fukutomi らは，ROS 分散環境においてノードを動作させるホストマシン上の計算資源を効率的に利用するための計算資源管理機構を提案した **ResourceManeger**．この研究では，計算資源使用率が一定に達したノードを動的に他のホストマシンに移行することで, 分散環境上でも効率的にノードを動作させることを可能としており，本研究とはノードの計算資源消費量を制限せずにノードの動的移行を行うという点で異なっている．しかし，Fukutomi らの実現手法ではノードの移行を行う際，ノードが保持しているデータのバックアップを取る必要があり, 手間がかかる．このことから, ROS 分散環境における計算資源管理機構の運用は, 計算資源消費量を監視して動的にノードを移行するよりも，計算資源の利用状況を事前に把握した上で管理する必要がある．

5 まとめ

本研究では，OSS ノードがシステムに与える影響を最小限に抑えることを目的とする．この目的を達成するために，各ノードの計算資源消費量を実行時に制限するサンドボックス機構を実装する．具体的には，Gazebo シミュレータ上でノードを動作させ，Linux の procfs と帯域幅監視ツールの一つである NetHogs を用いて計算資源消費量を記録し，これを基に計算資源消費量の見積もりを行う．見積もり結果より，コンテナ型仮想化機構の一つである cgroups と tc コマンドを用いて，ノードごとの CPU，メモリ，ネットワーク帯域の各計算資源における最大消費量に対して制限を課すことでサンドボックス機構を実現する．今後の課題として，ノードの計算資源消費量の見積もり手法がある．シミュレータ上でノードにどのような動作をさせるか，システムを動作させる期間をどうするかについて，システムを安全かつ効率的に動作させるための基準を設ける必要がある．ここで設けた基準が正確でなければ，サンドボックスを用いてシステムの計算資源消費量に上限を設けていても，本研究が期待する効果は得ることができない．基準について，どのようにして定めるのか，本研究が期待する効果を得るための基準とは何なのか考察する必要がある．また, 計算資源消費量を計測する間隔についても検討が必要である．

6 知能システムコースにおける本研究の位置づけ

知能システムコースでは，知能に関する課題および人と人工物の新たな関係性を構成論的な手法で追究する観点から、人の知的能力や機能の解明、数理モデル化、実世界への実装に関する具体的な課題に取り組み、その結果の評価を通じて、新しい方法論や、学問領域を切り拓く能力を育むことをカリキュラムポリシーとして掲げている．

本研究では，ROS という実世界への実装を補助する技術における課題を構成論的な手法で追究している．今後は実装を行い，サンドボックス導入前と導入後で各ノードの計算資源消費量がどう変化したか，またシステムの動作パフォーマンスがどう変化するかを評価する．

参考文献

[1] SoftBank：特集 | ロボット | ソフトバンク，入手先 <https://www.softbank.jp/robot/special/>（参照 2020-10-30）.

[2] SHARP CORPORATION：ロボホン，入手先 <https://robohon.com/co/introduction.php>（参照 2020-10-30）.

[3] Patrick Mihelich，James Bowman：ros-perception/image_common，入手先<https://github.com/ros-perception/image$_c ommon$>（参照 $2020-11-03$）.

[4] DiLuoffo,V., Michalson,R.W. and Sunar,B.：Robot Operating System 2: The need for a holistic security approach to robotic architectures，Proc.IJARS 2018，pp.1-15，SAGE Journals（2018）.

[5] 2014OpenSource Robotics Foundation：Gazebo，入手作<http://gazebosim.org/>（参照 2020-11-04）

[6] Fukutomi,D.，Azumi,T.，Kato,S.，et al.：Resource Manager for Scalable Performance in ROS Distributed Environments，Proc.DATE 2019，pp.1088-1093，IEEE（2019）.

[7] Michael Kerrisk：cgroups（7）-Linux manual page，man7.org，入手先 <https://man7.org/linux/man-pages/man7/cgroups.7.html>（参照 2020-10-30）.

[8] Docker Documentation|Docker Documentation, 入手先 <https://docs.docker.com/>（参照 2020-11-01）.