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： 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．このロボットを制御するシステムの開発において，ソフトウェアフレームワークの一種であるROSを用いる機会が増加している．ROSでは，ノードと呼ばれる機能モジュールを複数組み合わせることでシステムの構築を行う．ロボット制御システムの開発にROSを用いることで，OSSノードのような第三者が実装したノードをシステムに導入することが容易になり，開発効率の向上が期待できる．しかし，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．

キーワード：ROS, OSS, サンドボックス

Abstract: In recent years, the use of robots is expanding in various fields. The use of ROS, a type of software framework, is increasingly used in the development of robot control systems. In the ROS, the system is constructed by combining multiple functional modules called nodes. By using ROS in the development of robot control systems, it is easier to introduce nodes implemented by third parties, such as OSS nodes, into the system and thus improve development efficiency. However, the mixture of OSS nodes in a system makes it difficult to estimate the consumption of limited computing resources (CPU, memory, network bandwidth, etc.) in an embedded system. In this research, we propose a node sandbox to prevent unexpected load on the system by OSS nodes.

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, ノードと呼ばれる機能モジュールを複数組み合わせることでシステムを構築する．機能モジュールとは, 独立した機能を持つ部品のようなもののことを指す．ノードはLinuxにおけるプロセスに相当し, その多くはオープンソースソフトウェア (OSS) として公開されている．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かつ革新的であることから，現在のロボット開発には不可欠なフレームワークとなっている．ROSにはROS 1とROS 2が存在しており，ROS 2は単にROS 1の最新版として開発されているものではなく，あくまで別物として開発されている．ROS 2では，ROS 1で抱えていた問題点やROS 1で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．そのため，ROS 1で開発したシステムをROS 2に移行する作業があるため，現在のロボット開発ではROS 1を用いることが主流となっている．しかし, ROS 2にはサポートOSの増加や開発に使用するプログラミング言語のバージョン更新等があるため，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の最新バージョンであるROS 2を使用する．

2 課題

ロボットの高機能化に伴って, ロボット制御システムにおける脆弱性の報告が増えている．特に，多種多様なOSSノードの開発が進められて

いる [4]ROS では深刻な問題となる．ROS では，第三者が作成した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．特に，デバイス上で共有利用している計算資源である CPU，メモリ，ネットワーク帯域幅の消費量は，各ノードが必要とする計算資源消費量を事前に見積もることが難しい．加えて，現在の ROS システムでは，ノード毎の計算資源の管理を行うための機構が備わっていないため，ROS の分散環境におけるノードの割り当てが不均衡になり，計算資源を効率的に利用できない [5]．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である．図から，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ことがわかる．そのため，ロボット制御システム (ドローン) の動作が不安定になっている．このように，ロボット制御システムで利用可能な計算資源の量には上限があるため，ROS では，OSS ノードのような第三者が作成した計算資源消費量の予測できないノードが，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．

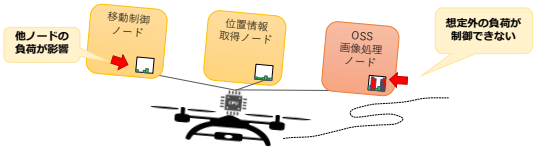

図 1 ロボット制御システムの例

3　提案する理論

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は，計算資源消費量を予測できないが故に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システム内に存在する各ノードが使用する計算資源量の見積もりを行う．次に，見積もりの結果を踏まえて，サンドボックスで制限する計算資源の設定を行い，各ノードに対して設定が反映されたサンドボックスを導入する．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 (ドローン) の動作が安定するようになる．

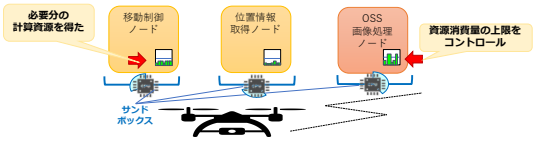

図 2 サンドボックス導入後のロボット制御システムの例

3.2　ノードの計算資源消費量見積もり手法

ノードの計算資源消費量の見積もりは，ノード動作中の各計算資源消費量を一定間隔で記録する機能を ROS フレームワークに組み込み，Gazebo を用いてシミュレーションを行うことで実現する．Gazebo とは，ROS で一般的に用いられるシミュレーションツールのことであり，これを用いることでシステムを模擬的に動かすことができる．この計算資源消費量の記録には，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表示に使用する．また，NetHogs は，プロセス毎のトラフィックを表示するものであり，ここではネットワーク帯域幅の表示に使用する．これらの出力結果の必要な部分のみを取りまとめたファイルを作成・保存し，一定間隔でデータの更新を行い，このファイルの内容を基に見積もりを行う．

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機構は，cgroup と tc コマンドを用いて，ノード毎の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実現する．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機構の一つであり，プロセスグループの計算資源の利用を制限・隔離する Linux カーネルの機能のことを指す．また，tc コマンドは，Linux カーネル内の通信を制御するものであり，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を指す．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 は徐々に実装を進めている段階であるため，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 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リソースコントローラーのことを指す．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 $\mathrm{net}_c ls$ 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の作成とする．

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[7] 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，高い機能性を持つ．しかし，Docker では，本研究において求めている以上に機能があるため，オーバーヘッドがかかる．そのため，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制限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 [5]．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いる．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には，計算資源の利用状況を逐一管理する必要があることがわかる．

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，ノード毎の 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，ノードを動かす期間をどうするかについて，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ため，最も重要な部分である．また，計算資源消費量の記録を更新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] 齋藤慶太, 森達哉：コンシュマー向けロボットの安全な運用に向けたセキュリティポリシー, コンピュータセキュリティシンポジウム 2017 論文集，Vol.2017，No.2，pp.1426-1433(2017).

[4] Computing Platforms Federated Laboratory：CPFL/Autoware_Toolbox, 入手先<https://github.com/CPFL/Autoware_Toolbox>(参照 2020-10-30).

[5] 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).

[6] 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).

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