シームレスなクラウド連携の実現に向けた軽量ロボットソフトウェア基盤 mROS 2-POSIX の予備評価

1020259 中村碧
指導教員：松原克弥

A Preliminary Evaluation of the Lightweight Robot Software Framework mROS 2-POSIX Toward Seamless Cloud Collaboration

Aoi Nakamura

概要： ロボットソフトウェア開発において，ROS の利用が増加している．クラウド連携を用いた分散ロボットシステムの構築が主流だが，ノード配置の柔軟性が課題となっている．特に，クラウドとロボットの CPU アーキテクチャの違いから，動的なノード再配置が難しい．先行研究での動的配置機構はオーバーヘッドの増加が問題とされ，mROS 2-POSIX を用いたアプローチも提案されているが，評価は異なるメッセージ型を送受信するアプリケーションに限定されている．
本研究では，mROS 2-POSIX を用いて ROS 2 の性能を評価．通信性能やメモリサイズを基に，動的配置機構の軽量ソフトウェア基盤として mROS 2-POSIX の評価を行う．

キーワード：ロボティクス,mROS2-POSIX

Abstract: In robot software development, the use of ROS is on the rise. While building distributed robot systems through cloud integration is popular, the flexibility in node placement remains a challenge. Specifically, the disparity in CPU architectures between the cloud and robots complicates dynamic node reallocation. Previous studies on dynamic placement mechanisms have reported increased overhead, and although approaches using mROS 2-POSIX have been proposed, evaluations have been limited to small-scale applications.
In this research, we evaluate the performance of ROS 2 using mROS 2-POSIX. Based on communication performance and memory size, we explore the potential of dynamic placement mechanisms.

Keywords: Roboethics, mROS2-POSIX

1 背景と目的

さまざまな産業向けやエンターテイメント関連のロボットシステム，自動車の自動運転技術や IoT システムの構築においても，これらのソフトウェア開発をサポートするフレームワークとして Robot Operating System（以下，ROS）の普及が増加している [1]．ROS のプログラミングモデルは，システムの各機能を独立したプログラムモジュール（ノード）として設計することにより，汎用性と再利用性を向上させて，各機能モジュール間のデータ交換を規定することで，効率的かつ柔軟なシステム構築を可能にしている．たとえば，カメラを操作して周囲の環境を撮影するノード，画像からオブジェクトを識別するノード，オブジェクトのデータを基に動作制御を実行するノードを連携させることで，自動運転車の基本機能の一部を容易に実装できる．

ROS のプログラミングモデルは，ロボット/IoT とクラウドが協力する分散型システムにおいても有効である．ロボットシステムのソフトウェア処理は, 外界の情報を取得する「センサー」, 取得した情報を処理する「知能・制御系」，実際に動作するモーターなどの「動力系」の 3 要素に分類できる [2]．クラウドロボティクス [3] において，主に知能・制御系のノードを高い計算能力を持つクラウドに優先して配置することで，高度な知能・制御処理の実現を促進できる．さらに，ロボットが取得した情報や状態などをクラウドに集約・保存することで，複数のロボット間での情報共有と利用を容易にする．一方で，現行の ROS 実装では，各ノードの配置をシステム起動時に静的に設定する必要があり，クラウドとロボット間の最適なノード配置を事前に設

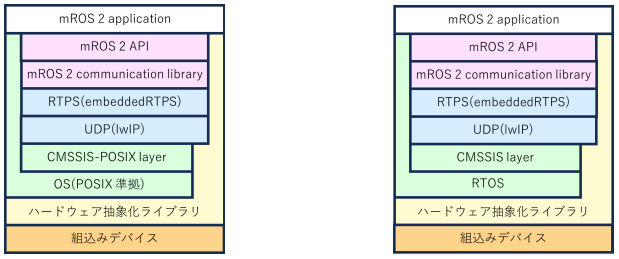
計する必要がある．しかし，実際の環境で動作するロボットは，ネットワークの状況やバッテリー残量の変動など，システム運用前に予測することが困難な状況変化に対応する必要があり，設定したクラウドとロボット間のノード配置が最適でなくなる可能性がある．このような状況変化への対応として，ノードを動的に再配置するライブマイグレーション技術があるが，多くの場合でクラウドとロボット間の CPU アーキテクチャが異なり，命令セットがそれぞれ違うため，実行中のノードをシステム運用中にマイグレーションすることは技術的に困難である．

菅らは，WebAssembly（以下，Wasm）を用いることで，クラウドとロボット間での実行状態を含む稼働中ノードの動的なマイグレーションする手法を実現した [4]．Wasm とは Web 上で高速にプログラムを実行するために設計された仮想命令セットアーキテクチャのことで，1 つのバイナリが複数のアーキテクチャで動作するため，異種デバイス間でのマイグレーションに適しているといえる．課題として，ROS 2 を Wasm 化したことでマイグレーション後のファイルサイズのオーバーヘッドが増大し，ノードの実行時間が大幅に増えてしまう問題が残った．柿本ら [5] は，組込みデバイス向けの軽量な ROS 2 ランタイム実装である mROS 2-POSIX を採用し，ROS ランタイムの Wasm 化にともなうオーバヘッド増加に対処した．しかし，採用された mROS 2-POSIX は指定したメッセージ型を ping-pong 通信するアプリケーション上でしか評価実験は行われておらず [6]，ライブマイグレーション後のオーバーヘッド増加を解決するロボットソフトウェア基盤として，ライブマイグレーションするアプリケーションが複雑化した場合の動作が明らかでない．

本研究では，クラウドとロボット間での実行状態を含む稼働中ノードの動的マイグレーションの実現に向けた mROS 2-POSIX の性能評価を行い，mROS 2 が ROS 2 と比べて動的配置機構の実現のソフトウェア基盤としてどの点で優位性があるのか明らかにすることを目指す．

2　mROS2-POSIX

mROS 2 は，ROS 2 ノードの軽量実行環境である．mROS 2 は，中規模の組込みデバイス上で事項されるプログラムに汎用デバイス上の ROS 2 ノードと自律的に通信する機能を提供している．このソフトウェア基盤によって，分散型のロボットシステムへの組込み技術の導入できる．組込みデバイスは計算資源が限定的であるが，リアルタイム性の向上および消費電力が削減をできる．そして，mROS 2 が POSIX[7] に対応したのが mROS 2-POSIX である．



(a) mROS 2-POSIX の内部構成　(b) mROS 2 の内部構成

図 1　mROS 2-POSIX と mROS 2 のソフトウェア構成

図 1（a）は，mROS 2-POSIX のソフトウェア構成を示す．mROS 2-POSIX アプリケーション層は，ユーザが実装する ROS 2 ノードに相当する．mROS 2-POSIX API 層および通信ライブラリ層は，ROS 2 の topic に相当する API および通信機能を提供する階層である．本階層は，ROS 2 のネイティブなクライアント通信ライブラリである rclcpp と互換性を保つように設計している．mROS 2 通信ライブラリでは，rclcpp のうち pub/sub 通信の基本的な機能のみ実装されている．利用可能な機能は制限されているものの，組込み技術を導入する ROS 2 開発者は，汎用 OS 向けのプログラミングスタイルを踏襲しながら C++によって mROS 2 のアプリケーションを実装できる．

Real Time Publish-Subscribe（以下，RTPS）プロトコルスタックには UDP でパブリッシャとサブスクライバ C++実装の embeddedRTPS[8] が採用されている．また，RTPS における SPDP および SEDP が実装されているため，通信の自立性が実現できる．UDP については組込み向けの C 実装である lwIP が採用されている．また，これらの階層は CMSIS-POSIX API に依存している．通信層の embeddedRTPS および lwIP は CMSAIS-POSIX に依存しており，図 1（b）に示す mROS 2 の CMSIS-RTOS を互換した層になっている．最下層にはハードウェアを抽象化

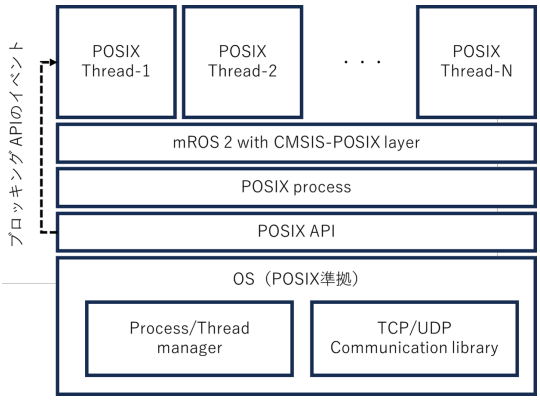


図 2　mROS 2-POSIX の実行方式

したライブラリがある．

mROS 2-POSIX の実行方式は，mROS 2 が対応しているリアルタイム OS との相違がある．リアルタイム OS では，組込みマイコンを実行資源の管理対象として，タスク単位でアプリケーションが実行される．POSIX においてはタスクに相当する概念はプロセスであり，そこから生成されるスレッドを実行単位として処理が進行している．

したがって，mROS2-POSIX は図 2 に示す実行方式を採用している．mROS 2-POSIX の実行単位であるノードは POSIX のスレッドに対応付けらており，mROS 2 の通信ライブラリ機能を担う実行資源は POSIX のプロセスとして捉え，これを管理対象としている．組込みマイコンでの通信処理におけるイベント割込みについては，POSIX 準拠 OS におけるブロッキング API の発行に相当させて処理されている．

mROS 2 と mROS 2-POSIX の機能構造の対応として，タスク管理機能は，スレッドの生成や開始などの POSIX 資源の管理機能に対応付けれている．メッセージキューおよびミューテックスによるスレッドの同期・通信および排他制御は，POSIX では Pthread によって対応している．メモリ管理機能は Alloc/Free で，時間管理は OS 時刻管理機能で対応している．lwIP における UDP パケット処理は，POSIX における OS 資源の操作に関する機能で実現されている．

3　評価方針

mROS 2-POSIX を評価するにあったって，mROS 2 と同様のアプリケーションを ROS 2 に実装し，比較評価する．評価項目としては，mROS

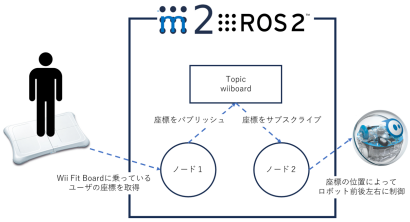


図 3　実装するアプリケーション

2-POSIX が ROS 2 と比べてどの点で優位性があるのかを明らかにするため，以下の項目を設定した．

- 通信性能の評価
- メモリサイズの評価

通信性能の評価については，1 回の通信を Wii Fit Board の値をサブスクライブするまでとし，それを 100 回行い，その平均と最大値と最小値と標準偏差を取ることで性能比較を行う．
メモリサイズの評価については，アプリケーションのバイナリを size コマンドで計測しそれぞれプログラムの実行可能コードを格納する text，初期化済みのグローバル変数と性的変数を格納する data，初期化されていないグローバル変数と性的変数を格納する bss の 3 分類を比較する．

4　評価アプリケーションの実装

評価に使用するアプリケーションは Sphero SPRK+と Wii Fit Board を用いたロボット制御アプリケーションである．アプリケーションは，Wii Fit Board はユーザの体重移動を検知し，Sphero SPRK+を前後左右に動かすことで，ユーザの体重移動に追従するように動作する．図 3 にアプリケーションの構造を示す．mROS 2-POSIX は，図 3 のようなハードウェアを通してロボットを制御するアプリケーションの評価は行われていない．したがって，本研究ではハードウェアを利用してユーザーがロボットを制御するアプリケーションを実装し，mROS 2-POSIX の評価を行う．アプリケーションは，Wii Fit Board に乗っているユーザの座標をパブリッシュするノードであると，Wii Fit BOard の座標があるトピックにサブスクライブし，受け取った値をもとに Sphero SPRK+を動作させるノードの 2 つによって実装されている．ROS 2 のアプリケーションは，Wii Fit Board の値を取得する OSS を ROS 2 ノード化したものと，Sphero

SPRK+を動作させることができるライブラリを提供している OSS を ROS 2 ノード化して通信させている．mROS 2-POSIX のアプリケーションの実装は ROS 2 のアプリケーションの機能と同じものを実装する必要があるため，同様のものを作成する．ROS 2 で動作するノードのソースコードはそのまま移植することはできないため，ROS 2 の rclcpp ではなく mROS 2-POSIX 固有の API を使用して作成する必要がある．また，Sphero SPRK+を動作させるライブラリは Python でコーディングされているため，C++のみに対応している mROS 2-POSIX ではそのまま使用できない．C++のライブラリに書き換える必要がある．

5 進捗と計画

現在の進捗状況として，アプリケーションは ROS 2 で動作するノードの作成が完了した．mROS 2-POSIX 対応は未実現である．mROS 2-POSIX では，Wii Fit Board の値をパブリッシュするノードは移植することができた．Sphero SPRK+のノードの移植は，ライブラリが Python で書かれているため, C++に書き換える必要があり, 現在書き換えに関する検討（調査）を行っている．

6 結言

本研究では, クラウドとロボット間での実行状態を含む稼働中ノードのライブマイグレーションの実現に向けた mROS 2-Posix の評価を行い, mROS 2 が ROS 2 と比べて動的配置機構の実現のソフトウェア基盤としてどの点で優位性があるのか明らかにすることを目指す．

動的配置機構を ROS 2 環境で実現した手法は，ライブマイグレーション後のファイルサイズのオーバーヘッド増加が課題であるとしている．課題解決のアプローチとして, ROS 2 の軽量ランタイムである mROS 2-POSIX を採用し，ライブマイグレーション後のオーバーヘッド全体を低減する試みを提案している．しかし，ROS 2 と mROS 2-POSIX の比較評価では，小規模なアプリケーションでしか評価が行われておらず, 実際に利用されるようなアプリケーション上での評価は行われていない．そのため，本研究では，ハードウェアを含めたユーザーがロボットを制御するアプリケーション上で ROS 2 と mROS 2-POSIX の通信性能やメモリサイズを比較評価し，mROS 2-POSIX を優位性を示す．また，mROS 2-POSIX が動的配置機構の軽量ランタイムとして，

7 知能システムコースにおける本研究の位置づけ

本研究は，ROS 2 の組込み向けデバイスでの実行環境である mROS 2-POSIX の評価を行うことで，クラウドとロボット間での実行状態を含む稼働中ノードの動的マイグレーションの実現に向けたソフトウェア基盤としての優位性を明らかにすることを目指している．その過程で，mROS 2-POSIX で動作するロボット制御アプリケーションの実装を行い，アプリケーションの動作を評価することで，動的配置機構を実装するソフトウェア基盤として適切であるか評価する．そのため，知能システムコースのカリキュラムポリシーに記載のある「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」に該当する．

参考文献

[1] ROSWiki：ROS/Introduction, url-http://wiki.ros.org/ROS/Introduction.

[2] ロボット政策研究会: ロボット政策研究会 報告書 RT 革命が日本を飛躍させる ,https://warp.da.ndl.go.jp/info:ndljp/pid/286890/www.meti.go.jp/press/20060516002/robot-houkokusho-set.pdf(2006).

[3] Kehoe et al. explored cloud-based robot grasping utilizing the Google object recognition engine, presenting their findings in the 2013 IEEE International Conference on Robotics and Automation, pages 4263-4270.

[4] 菅文人，松原克弥: クラウドロボティクスにおける異種デバイス間タスクマイグレーション機構の検討, 研究報告組込みシステム (EMB), Vol. 2022，No. 36，pp. 1-7(2022).

[5] 柿本翔大，松原克弥: クラウド連携を対象としたアーキテクチャ中立な ROS ランタイムの実現，情報処理学会研究報告， Vol. 2023-EMB-62，No. 51 ，pp. 1-7(2023).

[6] 高瀬英希，田中晴亮，細合晋太郎: ロボットソフトウェア軽量実行環境 mROS 2 の POSIX 対応に向けた実装および評価，日本ロボット学会誌，Vol. 2023-EMB-41，No. 8，pp. 724-727(2023).

[7] POSIX：https://www.ibm.com/docs/ja/zos/2.3.0?topic=ulero-posix

[8] A. Kampmann, et al.: “A Portable Implementation of the Real-Time Publish-Subscribe Protocol for Microcontrollers in Dis-tributed Robotic Applications,” Proc. of ITSC, pp.443 – 448，2019.