ロボットアプリケーションに向けたエッジサーバの資源管理手法の提案

福井誠人[1] 石綿陽一[2] 大川猛[3] 菅谷みどり[1]

概要：近年，さまざまな機能を持つコミュニケーションロボットが登場している．また，CPU の性能も向上しているとはいえ，ロボットなどでは利用できるハードウェアの性能が限られている．そこで高負荷な処理をクラウドサーバやエッジサーバにオフロードする研究が勧められている．本研究ではその中でもエッジサーバに着目してエッジサーバ上で実行されるロボットアプリケーションの資源管理の手法を提案する．エッジサーバで実行されるいくつかのロボットアプリケーションはアプリケーションごとに CPU を割り当てることで実行の効率をあげ応答時間を高めることを目指した．CPU の割り当てに関して単純なプログラムと Python による AI アプリケーションを使い評価を行った．

キーワード：ロボット，エッジコンピューティング，資源管理

1. 研究の背景と目的

近年，高機能なコミュニケーションロボットが広く普及している．高機能なロボットを実現するためには，高速な演算処理を可能とする高性能なプロセッサ必要となりサービスのシステム要件も高くなっている[1]．これに関して，CPU の性能も近年伸びている．しかし，一般的にロボットに搭載される CPU リソースは制限されていることが多い．限られた計算資源を使い，高機能なロボットサービスを提供するために高負荷な処理をクラウドにオフロードすることが考えられている[2]．しかし，ここにはクラウドとロボットの間の通信遅延がある．そこで，ロボットに近い位置に豊富な計算資源を持つエッジサーバを設置しエッジサーバにオフロードして計算を行うエッジコンピューティングが近年注目を集めている[3]．IoT デバイスからクラウドサーバに直接タスクをオフロードするのではなくエッジサーバを用いる研究がある[4]．この研究ではクラウドサーバを使う場合に比べてエッジサーバを使うことにより同等もしくはそれ以上の性能を達成していることを示している．

本研究ではエッジコンピューティング注目し，エッジサーバでの資源管理手法の提案をすることを目的として行った．特に，エッジサーバでのサービスとしてロボットと人とのコミュニケーションに関連した AI アプリケーションを想定してこれに向けたエッジサーバのプロセッサの割り当て手法を提案する．

2. 提案

本研究ではロボットと人とのコミュニケーションを対象とする AI アプリケーションに向けたエッジサーバのプロセッサの割り当て手法を提案することを目的として次の提案をする．(1) many-core CPU でのプロセッサの割り当て手法の検討，(2)小規模なサンプルアプリケーションによる many-core CPU の調査．(3)対象とするアプリケーションに対して任意のプロセッサを割り当てたときの性能評価．

3. many-core CPU でのプロセッサ割り当て手法

我々は下図のような複数の種類のロボットアプリケーションが稼働している状況を想定する．この図では Robot application program unit ごとにそれぞれ別のロボットアプリケーションが稼働している．それぞれのアプリケーションユニットではストリーミングデータ処理や制御計算，信号処理などいくつかの実行コンポーネントによって成る．それぞれのリソース要件はアプリケーションに依存して異なる．ここでユニット 1 がユニット 0 と比べて 2 倍の処理量があるとしたときに 2 倍のプロセッサリソースをユニット 1 に割り当てることで円滑に処理することを検討する．

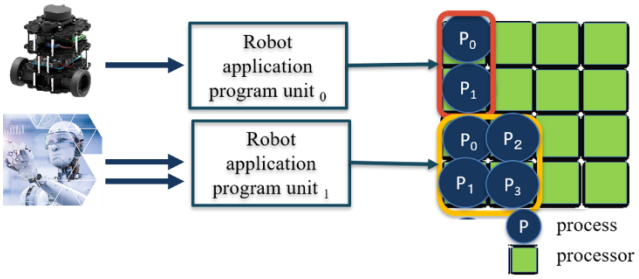

図 1 想定するロボットアプリケーション

4. 予備実験

AI アプリケーションを想定したプログラムを動かす前に単純な CPU を消費するプログラムを many-core CPU で動かしたときの CPU リソース割り当て量と実行時間につ

1 芝浦工業大学
Shibaura Institute of Technology
2 VA Linux Systems Japan 株式会社
VA Linux Systems Japan
4 東海大学
Tokai university

いての調査を行った．複数回 fork() することで子プロセスを作るマルチプロセスプログラムに割り当てるプロセッサ数を変化させながらプログラム全体の実行時間を調査した．結果を次の図に示す．

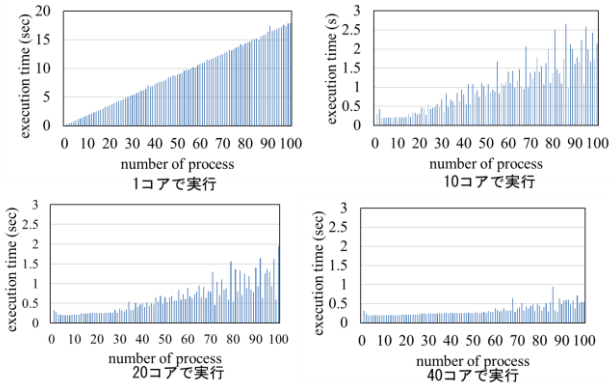

図 2　予備実験結果

ここではそれぞれの子プロセスでは多数のループを伴う円周率の計算を行い CPU を消費しているため，プロセス数を増やすにつれ実行時間は増加した．このプログラムに対して割り当てる CPU を増やしていくことで反比例的に実行時間は減少する．本実験では単純な fork() を複数回呼び出すプログラムであるがこのようなプログラムが複数回呼び出されるソフトウェアについて考える．ここで，本結果をもとに次の一般式を導いた．

$$(\textit{実行時間の推測値})\ T_{exec} = \sum_{i=0}^{jobs}\left(\alpha_i \times \frac{Process_i}{Core_i} + \beta_i\right)$$

数式 1　実行時間の推測式

ここで，α，β はそれぞれプログラムによって定まる定数であり jobs はそのソフトウェアを構成する個々のアプリケーションユニットの個数である．

5. 想定アプリケーションに対する CPU 割り当てと実行性能の調査

本研究の目的としてロボットと人のコミュニケーションに向けた AI アプリケーションをエッジサーバで効率的に実行することがある．目標とする応答性を達成するために計算負荷を推測して適切な CPU を割り当てることが必要となる．これを実現するための最初の段階として Python で書かれた AI アプリケーションを任意の CPU に紐づけて実行しこのときの性能について調査した．Python でのプログラムは sklearn ライブラリを使い並列で処理できるようにしたものを使用した．今回作成したプログラムは次の動作を行う．fork() したのちに execve() コマンドを使い先ほどのプログラムを実行する．このときに sched_setaffinity() を使い指定した CPU を割り当てることを可能にした．

6. 結果

実験は表 1 のマシンを使い行った．このときの結果を図 2 に示す．割り当てたプロセッサを 1 から 2 に増やしたとき実行時間は 2 倍以上増加しその後 5 プロセッサまで増やしたときにわずかに増加しその後はおおむね横ばいとなった．当初の予測では割り当てるプロセッサ数を増やすことで実行時間を短縮できると考えていたが，それに反して実行時間が増加する結果となった．現段階では要因の特定ができていないため今後の課題としたい．

表 1　使用したコンピュータ

システム	Fedora32 Python 3.8.5
CPU	Intel Core i7-7700 4 コア 8 スレッド
メモリ	16GB

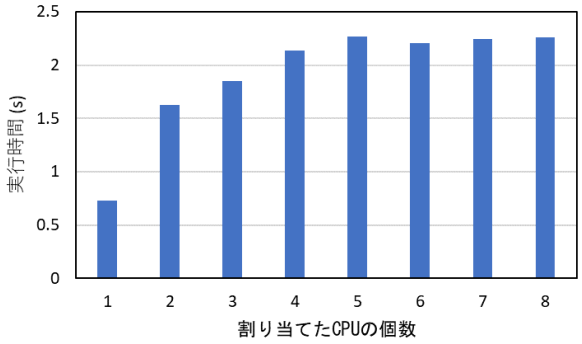

図 3　割り当てた CPU と実行時間の関係

7. まとめと今後の課題

本研究ではエッジサーバの資源管理手法の提案を目的として many-core CPU でのプロセッサの割り当ての手法を検討し，2 つのアプリケーションからその方法について調査した．並列化が容易であると考えられるプログラムでは CPU リソースの割り当ての優位性があることが分かったが想定する Python アプリケーションでは期待通りの動作ではなかった．これの要因を特定するとともに提案する割り当て手法を使ったシステムの実装，評価を今後行う．

参考文献

[1]“Artificial Intelligence and Robotics”, https://arxiv.org/ftp/arxiv/papers/1803/1803.10813.pdf (参照 2020-11-5)
[2]Sandeep Chinchali, ”Network Offloading Policies for Cloud Robotics:a Learning-based Approach” , arXiv:1902.05703v1
[3]Youdong Chen and Qiangguo Feng, “An Industrial Robot System Based on Edge Computing: An Early Experience”, HotEdge’18
[4]R. M. Shukla and A. Munir, "An efficient computation offloading architecture for the Internet of Things (IoT) devices," 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2017, pp. 728-731, doi: 10.1109/CCNC.2017.7983224.